INTERPROCESSOR COMMUNICATION LINK

AN INTERPROCESSOR COMMUNICATION LINK

FOR

DATA GENERAL MINICOMPUTERS

By

MICHAEL EDWARD BRETT, B. SC.

A Project Report

Submitted to the School of Graduate Studies

in Partial Fulfilment of the Requirements

for the Degree

Master of Engineering

McMaster University

May 1977

MASTER OF ENGINEERING (1977)
(Engineering Physics)

McMASTER UNIVERSITY
Hamilton, Ontario.

TITLE: An interprocessor communication link for Data General minicomputers

AUTHOR: Michael Edward Brett, B.Sc. (Brock University)

SUPERVISOR: Professor T. J. Kennett

NUMBER OF PAGES: iv, 65

ABSTRACT

The ACTR (Asynchronous Communications Transmitter Receiver) is a serial data transfer link for the Data General ECLIPSE and NOVA minicomputer lines. The ACTR allows the interconnection of computers in the NOVA and ECLIPSE lines into a multiprocessor system by permitting blocks of data to be transferred through the computers' program I/O facitities. Such a small computer multiprocessor system is a powerful, high flexible alternative to a single large computer in many applications. The major application of the ACTR is in systems where the linked processors are either far remote from one another or where the system is so configured that a master/slave environment is practical.

This report will deal with the theory of Operation of the hardware as well as the software control of the ACTR. A method of handling the ACTR in a multi-tasking environment under the Data General operating systems, RDOS/RTOS, will also be developed.

ACKNOWLEDGEMENTS

The author would first like to thank my supervisors, Dr. T. J. Kennett, for giving me a sound base in minicomputer theory and operation and the confidence to continue on in this line into the industrial world. I would also like to acknowledge the assistance of Gord Cormick and Kenrick Chin who combined to teach me a great deal about the practical side of minicomputers and digital electronics.

Special thanks are due to Litton Systems (Canada) Limited for the use of their ECLIPSE S/200 computer system in the development of some of the software presented in this report.

Finally, I would like to extend my appreciation to the Engineering Physics Department of McMaster University for both accepting me as a graduate student and providing me with a very challenging and rewarding course of study.

TABLE OF CONTENTS

		Page
ABSTRACT		i
ACKNOWLEDGEMENTS		ii
TABLE OF CONTENTS		iii
LIST OF FIGURES		iv
CHAPTER		
1	INTRODUCTION	1
2	THEORY OF OPERATION	6
3	PROGRAMMING THE ACTR	19
4	CONCLUSIONS	31
APPENDIX A	ACTR MASTER-SLAVE HANDLING ROUTINES	33
APPENDIX B	ACTR DIAGNOSTICS	51
BIBLIOGRAPHY		65

LIST OF FIGURES

Figure		Page
1.	ACTR Transmitter	7
2.	ACTR Receiver	8
3.	ACTR Device Flags	9
4.	Transmitted Pulse	11
5.	Transmitter Clocking	13
6.	Receiver Clocking	14
7.	Interrupt Timing	17
8.	I/O Instruction Format	20
9.	ACTR Control Summary	29
IO.	ACTR Master-Slave Message Format	34

CHAPTER 1

INTRODUCTION

The most rapidly growing use of the world's communicatoin links is for data transmission. The most rapidly growing area in the exploding data processing industry is teleprocessing. The reason is the versatility that the interlinking of computers can bring, plus the potential benefits to the individual of having this computing power at his fingertips. The phenomenal growth of the minicomputer industry in the last decade has been highly influential in the expanding field of data transmission.

In situations characterized by relatively simple mathematical operations and substantial data communication and formatting requirements, a minicomputer multi-processor is often less expensive and far more flexible than any single medium to large scale computers capable of meeting all the job requirements.

A minicomputer multi-processor system is usually characterized by a large central computer, complete with mass storage and hard copy devices, which is connected to a number of smaller, remote processors. The remote systems can control or monitor some process or series of processes continually, as the central computer stands ready to assume a remote system function in the event of a failure of a remote CPU. While in this backup mode, the main computer can be employed on lower priority tasks such as data analysis or data collection from the remote processors.

In critical real-time situations, this redundancy may give a measure of safety to a total system, assuring continued operation even when a catastrophic failure occurs in a major system component.

In order to implement a multi-processor system of any description, a reliable method of processor-processor communication is needed. In the Data General NOVA and ECLIPSE minicomputer lines, data transfer links can be divided into two classes, with each class characterized by the method of Input/Output that it employs. The first class involves all devices that transfer information through the use of the data channel facility. A device connected to the data channel can, at its own request, gain direct access to memory, using a minimum of processor time. The second class involves all devices that process data through program I/O (all imput/output through accumulators). Handling data transfers between external devices and memory under program control requires an interrupt plus the execution of several instructions for each word transferred.

Data General's Multiprocessor Communications Adapter (MCA) is a data transfer link of the data channel variety. The MCA facilitates the interconnection of up to fifteen computers in the NOVA and ECLIPSE lines into a multiprocessor system by permitting blocks of data to be transferred at high speeds from one computer to another. Data rates are primarily determined by the processor's channel facilities. Typical data rates for a single link range from 70 KHz for a pair of NOVA computers to 140 KHz for NOVA line computers with the high speed data channel feature.

This report will deal with the implemention of an asynchronous data link of the program I/O variety for Data General NOVA and ECLIPSE minicomputers. The Data General Corporation NOVA and ECLIPSE lines of computers are general purpose, four accumulator, stored-program conputers with a word length of 16 bits. The basic instruction set contains instructions that perform fixed point arithmetic between accumulators; transfer of operands between accumulators and main storage and logic operations between accumulators. In addition to an assembler and a macroassembler, there are higher-level language processors available which include ALGOL, BASIC, FORTRAN IV and 5. There is a wide array of operating systems available for the NOVA/ECLIPSE line of minicomputers. These range from the Stand-Alone Operating System (SOS) to the Real-time Disc Operating System (RDOS).

The ACTR (Asynchronous Communication Transmitter Receiver) is a self-clocking data transfer unit which consists of independent transmitter and receiver sub-sections. Information is clocked bit by bit onto a transmission line. A complete word transfer consists of the movement of 20 bits: a 16 bit data word and 4 control (status) bits. Each transmitted bit is accompanied by a clock pulse which is used by the receiver in the other CPU to indicate the data strobe cycle. No computer intervention is needed in the receiving system until all 2Ø bits are received and shifted into place in the receiver buffers. For a cable length of 20 feet, a full 2Ø bits can be strobed into the receiver buffer in 8.0 micro-seconds. With a minimal ACTR handler (no error checking)

in both processors, typical data rates would be approximately 26 kHz for a NOVA 1210 and 37 KHz for a ECLIPSE S/200.

The MCA is not required in systems with only modest intercommunication requirements. In such systems, an interconnection of processors using an asynchronous interface is simpler and less expensive. The MCA cannot be used in systems where the linked processors are not contained in the same frame as the transfer distance for a MCA cannot exceed 15 feet. The ACTR has adjustable transfer clocks which allow the respective transmitter and receiver to be tuned to allow for the type and length of transmission line between them. The MCA achieves its much greater transfer rate by employing sufficient lines to move whole words at a time and provide all control signals. The ACTR on the other hand requires only two interconnecting cables and employs line drivers and receivers on each cable to increase the transmission range far beyond the limit of the MCA. Since clock pulses are transmitted with each datum bit, synchronization between the two computer systems is not a problem.

As the distance between the processors increases, the greater the chance of reception errors occuring. The four control bits of the ACTR can be used to supply transmission parity checks as well as STX (start of text) and ETX (end of text) capabilities to the user.

The major use of the ACTR is for data transmission in systems where time is not a crucial factor or where the interconnected processors are too remote to use the faster MCA. Such

a system could consist of a remote processor controlling a Multi-channel Analyzer. Periodically an energy spectrum would be transferred from the remote CPU to a central processor with mass storage capabilities(moving or fixed head disc, magnetic tape, paper tape etc.). This transfer of data would take place during the times that the remote system was not actively involved in a monitoring activity so that no new information is lost. The central processor could handle a number of such remote systems and provide an economical means of recording information by eliminating the need for storage facilities on each remote system.

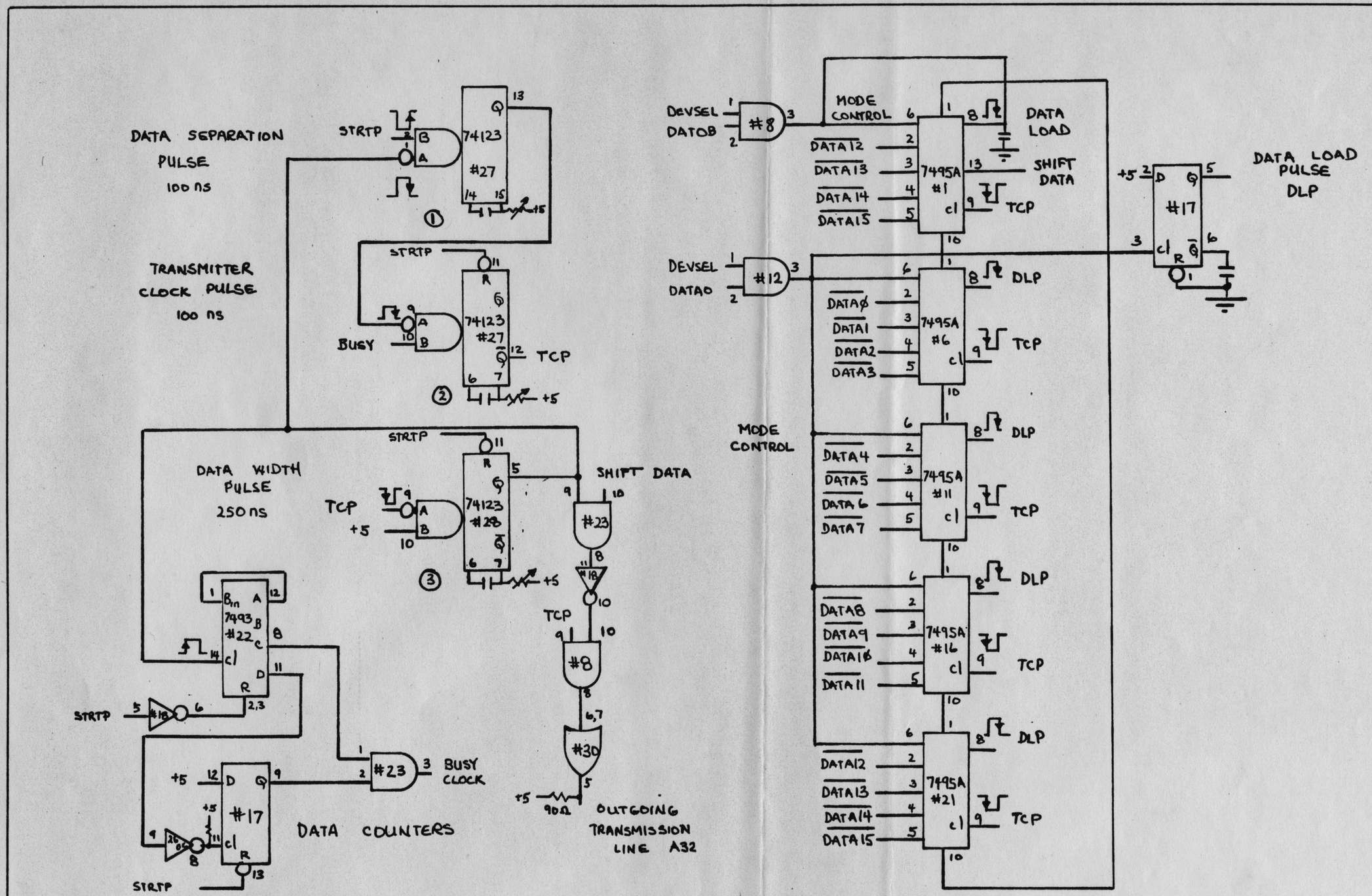
Chapter 2 of this report will deal with the hardware aspects of the ACTR by outlining the clocking sequences of the transmitter and the receiver and the interface of the ACTR to the computer I/O hardware, In chapter 3, I will be discussing the software control of the ACTR and the linkage of the ACTR into the Data General Operating Systems, RDOS and RTOS. Chapter 4 serves as a summary of ACTR and its place in data communication between Data General minicomputers. Appendix A is a handling routine for the ACTR in a master-slave multi-tasking environment and Appendix B is a hardware diagnostic for the ACTR.

CHAPTER 2

THEORY OF OPERATION

The ACTR is an asynchronous self-clocking data transfer link that conforms to standard Data General interface logic in all but its use of the Busy and Done flags. The ACTR Done flag is used to indicate the state of the receiver section while the Busy flag reflects the current condition of the transmitter. The ACTR is attached to the I/O bus of the computer and is connected to the ACTR of the other computer by dual transmission lines. Although mounted on a single circuit board, an ACTR contains independent receiver and transmitter and receiver subsections. The complete ACTR assembly is shown in Figures 1-3.

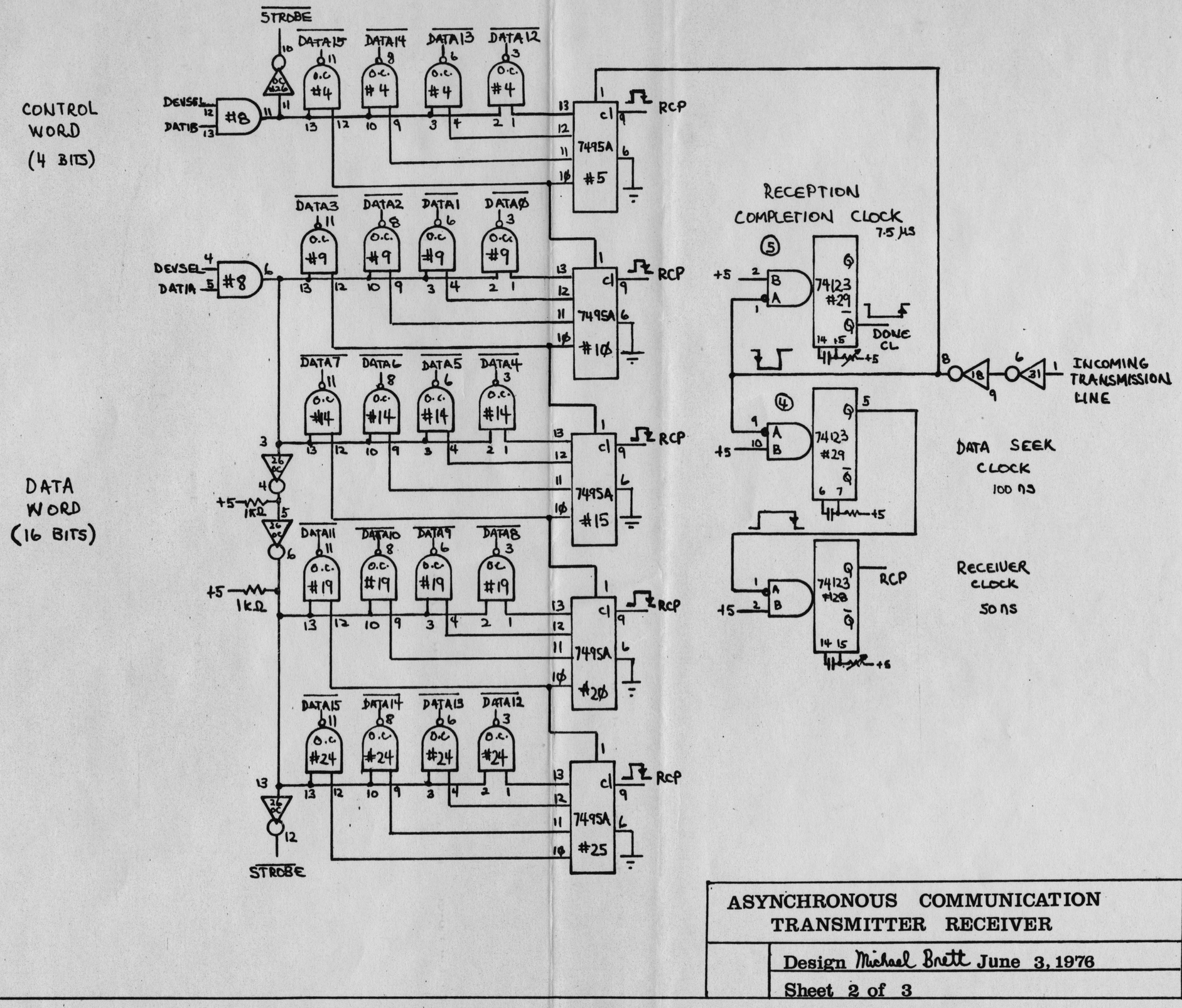
Each data word is sent under program control. A complete transmission consists of the transfer of 20 bits; 16 data bits (DG word) and 4 control bits (transmission status bits for program use). After data bits are strobed into the transmitter buffers, the data transfer is initiated by setting the Busy flag. After the transmitter has sent all 20 bits, it clears the Busy flag. The transmitter subsection cannot request an interrupt when it has completed its operation, Accompanying each transmitted bit is a clock pulse which is used to activate the clocking cycle of the ACTR receiver in the other system. When the reception of the incoming word is completed, the receiver sets the Done flag. The ACTR receiver is connected to the interrupt system (i.e. the ACTR receiver can generate an interrupt request)



ASYNCHRONOUS COMMUNICATION TRANSMITTER RECEIVER	
	Design Michael Brett June 3, 1976
	Sheet 1 of 3

ACTR TRANSMITTER

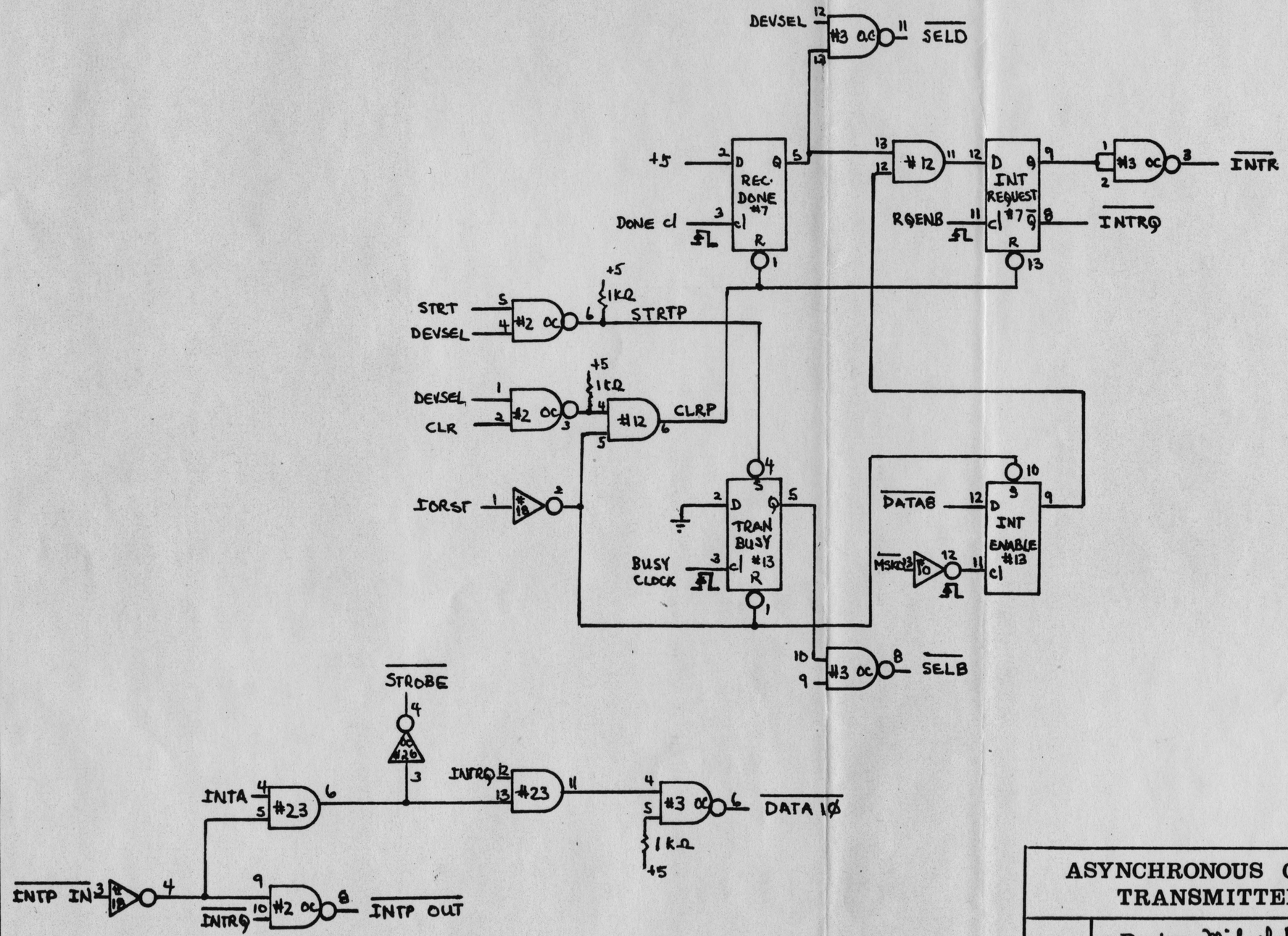
Figure 1



ASYNCHRONOUS COMMUNICATION TRANSMITTER RECEIVER	
	Design Michael Brett June 3, 1976
	Sheet 2 of 3

ACTR RECEIVER

Figure 2



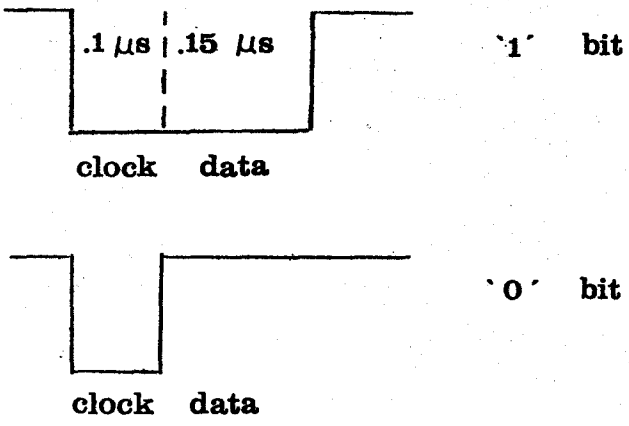
ASYNCHRONOUS COMMUNICATION TRANSMITTER RECEIVER	
	Design Michael Brett June 3,1976
	Sheet 3 of 3

ACTR DEVICE FLAGS

Figure 3

and can respond to an INTA instruction by returning its device code. When the processor realizes that a new data word is present in the receiver, the information is strobed into specified accumulators under program control. To allow the processor to sense the state of the ACTR, it places its Busy and Done flags on the SELB and SELD lines whenever it recognizes its device code.

The transmitter buffers are loaded from the I/O bus by the DATOA (data word) and DATOB (control bits) signals respectively. The transmission of data is controlled by 3 "clocks" (retriggerable monostables); the Data Separation pulse, the Transmitter Clock pulse and the Data Width pulse. The Data Separation pulse, as its name suggests, is used to separate the individual data bit transmissions; to provide the receiver hardware time to act on each one. This pulse is initially triggered by the STRT signal from the processor. The STRT signal also sets the Busy flag (which remains high as long as the transmission is in progress). It is assumed that when the STRT signal is issued by the processor, the information to be transmitted is already in the transmitter buffer. On the falling edge of the Data Separation, the Transmitter Clock pulse is triggered. This pulse is used to shift the next bit into position to be transmitted. The rising edge of the Transmitter Clock generates the Data Width pulse. This pulse is ANDed with the datum bit to be sent to produce a pulse of a specified width. This pulse is then combined with the clock pulse and placed on the Transmission line. Figure 4 details the actual pulse that is transmitted. Due to



Times shown are characteristic of a 20 ft. co-axial cable

Negative clock pulse is sent for greater noise immunity

TRANSMITTED PULSE

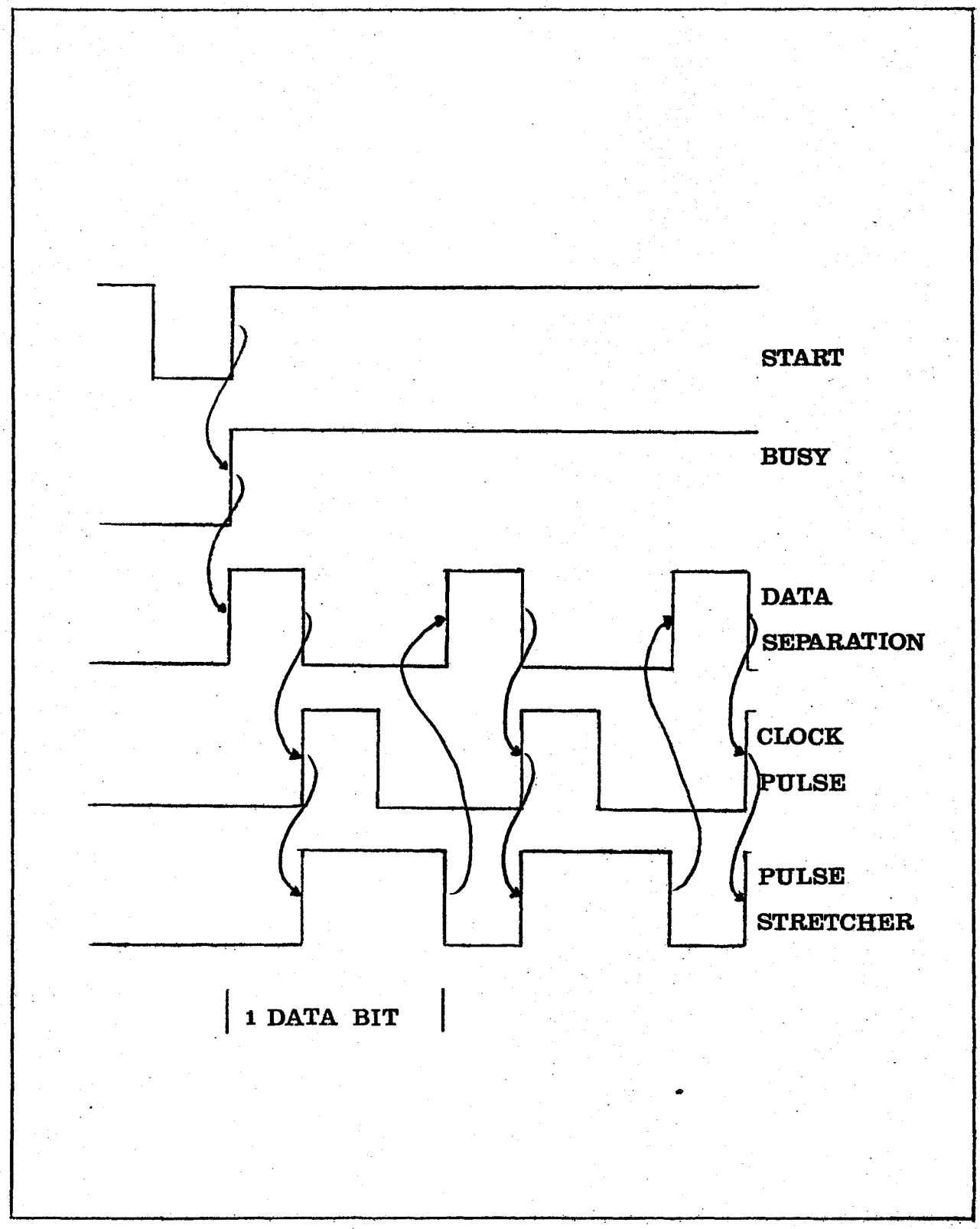
Figure 4

the manner in which the incoming information is clocked into the receiver buffer, the Data Width must be at least a factor of two longer than the Transmitter Clock.

The falling edge of the data width pulse increments a counter network which is responsible for terminating the clocking cycle at the end of the transmission. When the counter reaches 2Ø, the Busy flag is clocked over to Ø. When Busy is in the Ø state, the Transmitter Clock pulse cannot be generated on the falling edge of the Data Separation pulse and the clocking sequence is halted. The counter is zeroed at the start of each transmission by the STRT pulse. The complete sequence for the transmitter is outlined in Figure 5.

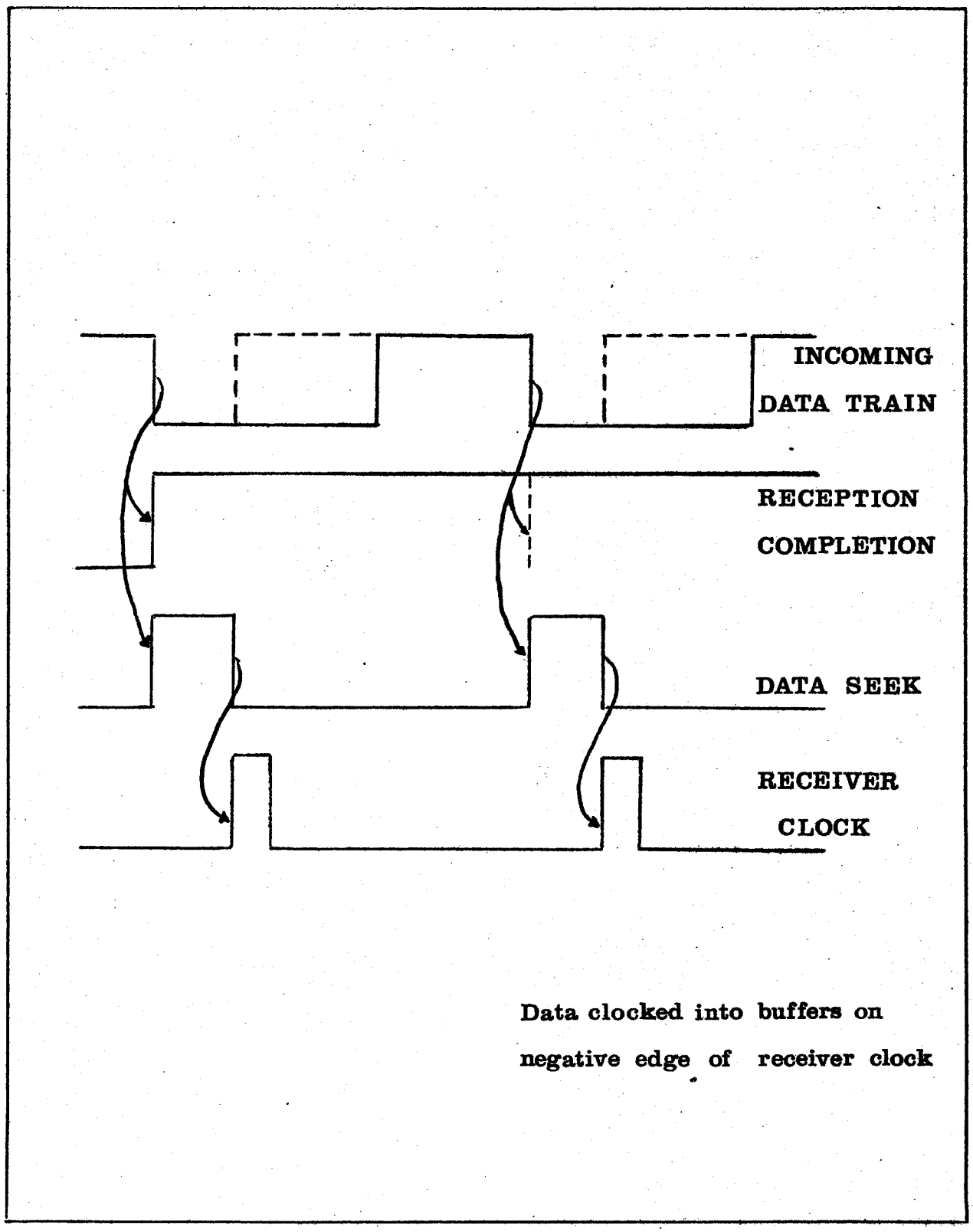
The receiver is activated on the falling edge of the incoming datum pulse (the negative of the Transmitter clock). This edge triggers the Data Seek and Reception Completion clocks. The Data Seek pulse must remain high for more than half the length of the Data Width of the transmitting ACTR. The Data Seek is used to indicate when the actual transmitted datum is incoming (i.e. when the clocking signal that begins each transmission is completed). The falling edge of the Data Seeks generates the Receiver Clock Pulse which actually shifts the incoming datum bit into the receiver buffer.

The Reception Completion clock is triggered on every incoming transmission. This clock is set to fall in a time that corresponds to approximately 1½ times the period of a complete (1 bit) transmission. Each incoming data transmission will re-trigger this clock (put the clock cycle back to the beginning)



TRANSMITTER CLOCKING

Figure 5



RECEIVER CLOCKING

Figure 6

so that the Reception Completion clock will not fall until the transmission of all 20 bits is completed. The entire clocking sequence of the receiver is illustrated in Figure 6.

The clocking times shown in Figures 1 and 2 are representitive of a system with a cable length of 20 feet. These times will vary with the length and type of cable used to connect the two processors. However, they do give a base point at which to start tuning an ACTR pair.

When the Completion clock falls, it clocks the DONE over to 1. This condition can be determined by sensing the DONE flag or through the interrupt, standard on the NOVA (ECLIPSE) line. Since a reception of data can normally occur at any time, the normal operating environment for a system using the ACTR would be with the interrupts enabled, so that the CPU can respond to a data reception whenever it happens.

In every instruction cycle, the processor generates RQENB which places the interrupt request signal INTR on the bus from a given device (i.e. sets its INT REQ flip flop) if its Done flag is set and its Interrupt Disable flag is clear. Under program control, the processor will generate the MSKO signal to set up the Interrupt Disable flags of all devices according to the information on the data lines (DATA8 = 1 to set the ACTR Interrupt Disable and DATA8 = 0 to clear the ACTR Interrupt Disable, In the ACTR interface, the actual flag is a INT ENABLE but it responds to a generated MSKO instruction as any standard DG interface would).

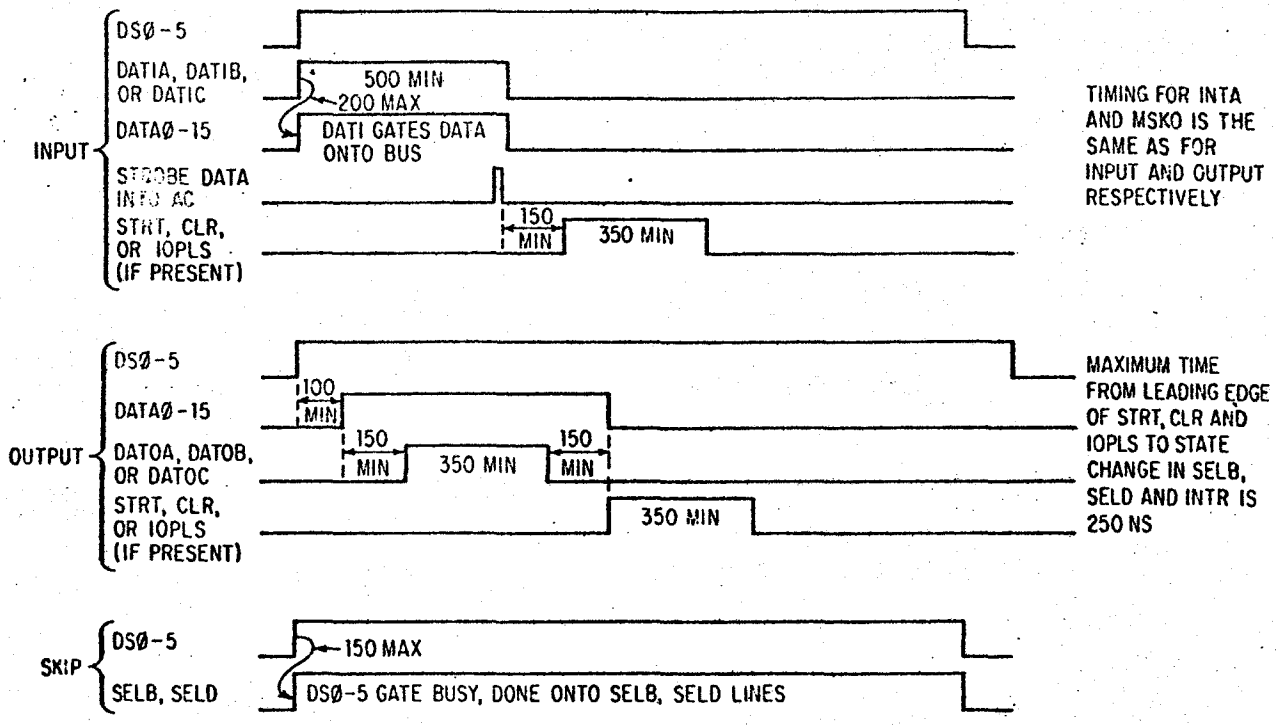
After an interrupt has been recognized by the processor, program control will go to the interrupt handler specified in

memory location 2 of the processor. The interrupt handler can determine which device needs service by sensing Done or it can issue INTA to read the device code of the nearest device requesting an interrupt.

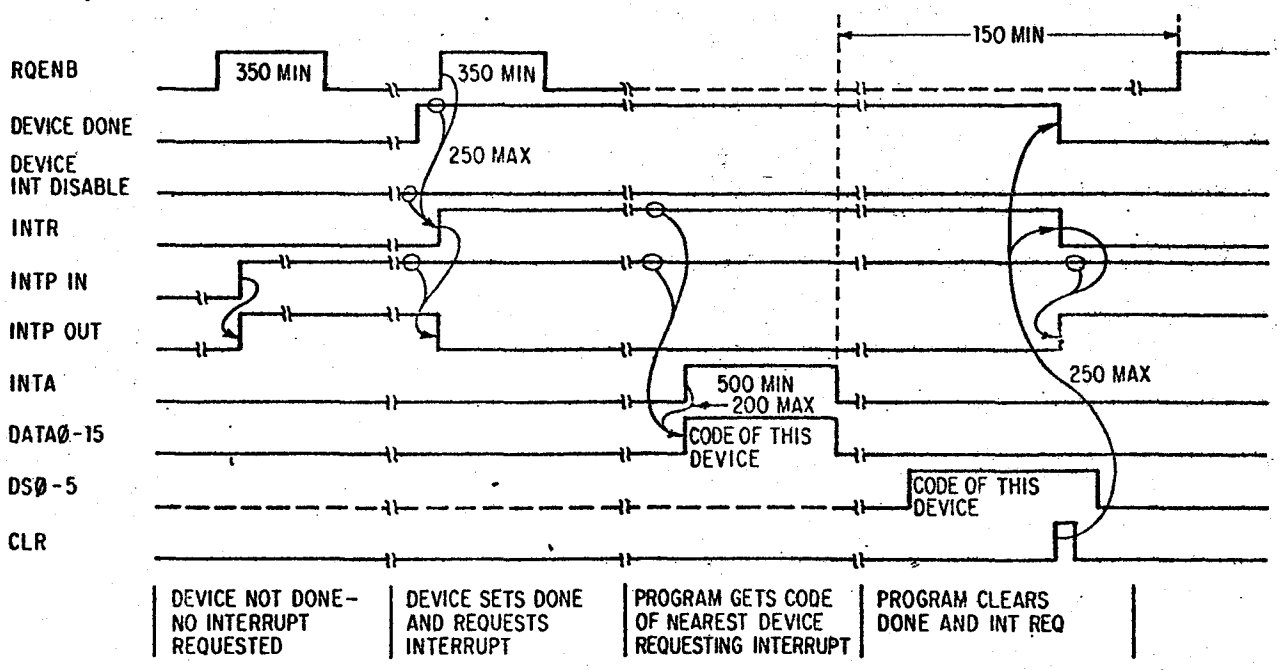
When an INTA instruction is executed by the processor, the INTP IN signal is generated. The ACTR will receive this signal only if no INT REQ flip flop is set in a device closer to the processor on the bus; the INTP must be terminated at the first device where the INT REQ is set. A second signal, INTA, is generated at a time sufficiently long afterwards to ensure that the INTP signal has been terminated. The INTA signal is used by the device, which terminated the INTP, to clock its device code onto data lines 10-15 of the I/O bus. This data is strobed into the processor at the end of the INTA. The timing of the logic signals that control the interrupt sequence is outlined in Figure 7. Once execution of the INTA instruction is completed, the device code received can be used to vector control to the appropriate routine to service the interrupt.

The interrupt service routine for the ACTR must strobe the received data into the processor by issuing a DATIA (data word) and DATIB (control bits) signal over the I/O Bus. The interrupt handler must clear Done by issuing a CLR signal so that the ACTR will not immediately request another interrupt for the same reception when the interrupt system is turned on and Interrupt Disable is cleared. Clearing Done also clears INT REQ, disabling INTR.

The timing of the various clocks in the ACTR is controlled by resistor (trimpot)- capacitor pairs. The clocking systems



PROGRAMMED TRANSFERS (IN-OUT INSTRUCTIONS)



PROGRAM INTERRUPT TIMING

Figure 7

in the two connected ACTR interfaces must be tuned to one another by adjusting the appropriate trimpots so as to obtain the maximum data transfer rate possible, given the characteristics of the connecting transmission lines and the latency of the interface elements.

The next chapter will deal with the software aspects of data transfer with the ACTR. The I/O structure of the NOVA (ECLIPSE) minicomputers will be reviewed with special attention to the program control of ACTR.

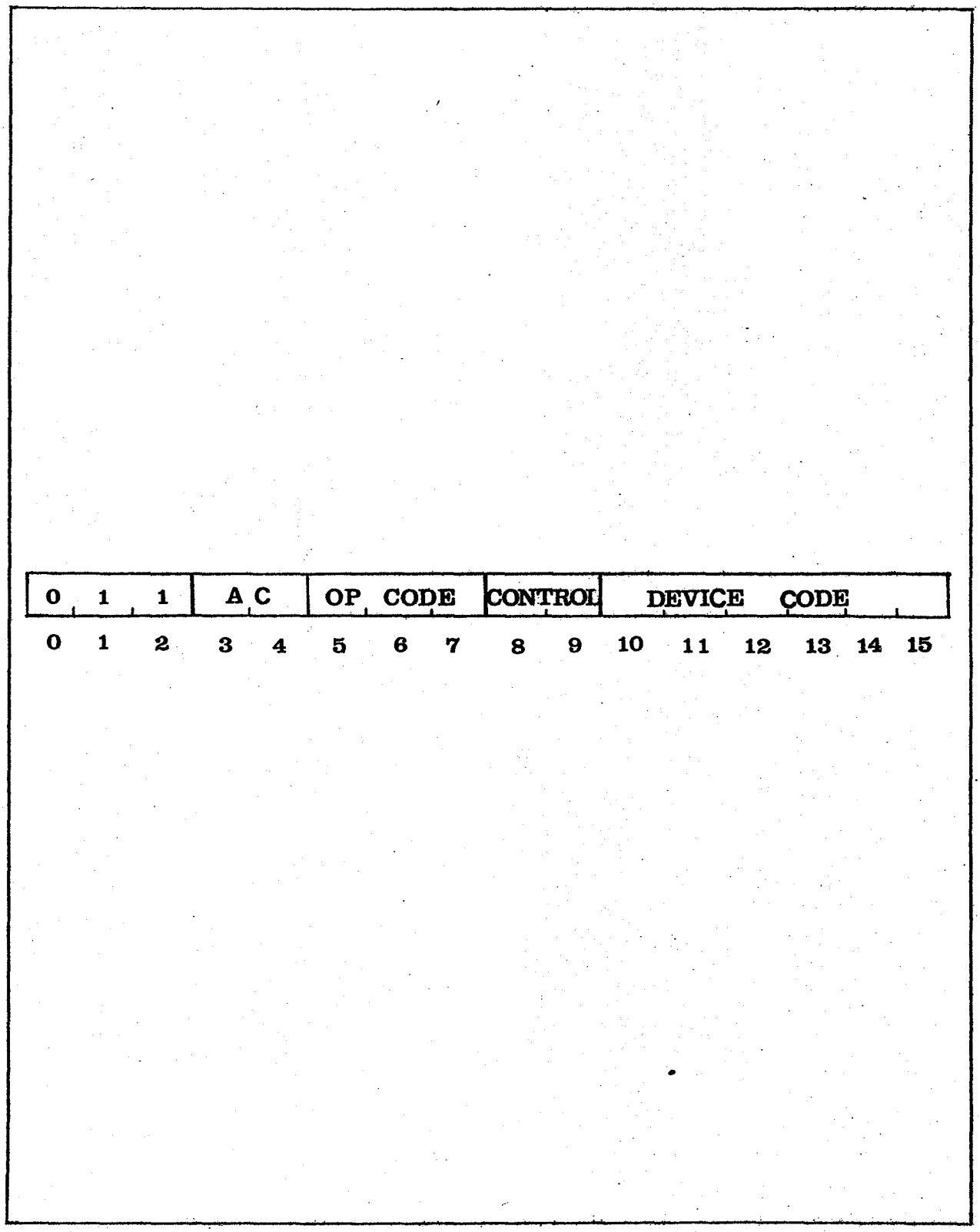
CHAPTER 3

PROGRAMMING THE ACTR

The software control of the ACTR is very straight forward but to be able to use this device effectively, the I/O handling facilities of the NOVA (ECLIPSE) must be well understood. This chapter will be devoted to a detailed description of the programming concepts of the I/O system with specific references to the handling of the ACTR.

In order for the processor to perform useful work for the user, there must be some method for the program to transfer information outside the machine. The Input/Output (I/O) instruction set provides this facility. There are eight I/O instructions which allow the program to communicate with I/O devices, control the I/O interrupt system, control certain processor options and to perform certain processor functions.

The NOVA (ECLIPSE) line has a 6-bit device selection network, corresponding to bits 10-15 in the I/O instruction format. Each device is connected to this network in such a way that it will only respond to commands with its own device code. Each device also has two flags, Busy and Done, which control its operation. In conventional DG devices, when Busy and Done are both zero, the device is idle and cannot perform any operations. To start a device, the program must set Busy to 1 and set Done to 0. When a device has finished its operation, it sets Busy to 0 and Done to 1. The ACTR, being essentially two devices in one



I/O INSTRUCTION FORMAT

Figure 8

(transmitter and receiver), does not conform to the standard DG flag configuration. To begin a transmission, the ACTR Busy flag must be set. When the transmission is completed, the ACTR will clear the Busy flag. When a reception is completed, the ACTR sets its Done flag.

The format for the I/O instructions is illustrated in Figure 8. Bits Ø-2 are Ø11, bits 3-4 specify the accumulator, bits 5-7 contain the operation code, bits 8-9 control the Busy and Done flags in the device, and bits 10-15 specify the code of the device. The six bits provided for the device code in the I/O format mean that 64 unique device codes are available for use. Some of these device codes, however, are reserved for the CPU and certain processor options. Most of the remaining codes have been assigned to particular device by Data General. The device code normally chosen for the ACTR is 40_8. In standard DG hardware, this corresponds to a Synchronous Communication Receiver (SCR), a device which will not be used on a system with an ACTR.

In order to operate the ACTR transmitter, the program must first ensure that the device is not currently performing some operation. This is done by testing the Busy flag for a Ø state by

```
SKPBZ  ACTR      ; IS BUSY ZERO?
JMP. -1          ;NO...CONTINUE CHECKING
                 ;ACTR NOT BUSY
```

Next the control and data words must be loaded into the transmitter buffer. The control bits can be used to inform the receiving

computer of the nature of the data transfer. (i.e. start of message, end of message, data acknowledgement etc.). After loading the data into the interface, a start pulse must be sent to the hardware to begin the actual transmission (set Busy to 1).

```
DOA ACS , ACTR      ; LOAD ACTR BUFFER A WITH
                    ; DATA FROM ACS
DOB ACD , ACTR      ; LOAD ACTR BUFFER B WITH
                    ; CONTROL BITS FROM ACD
NIOS      ACTR      ; NO I/O TRANSFER
                    ; -S- SETS BUSY TO 1 TO
                    ; BEGIN TRANSMISSION
```

Since the start pulse can be issued as soon as the hardware buffers are loaded, this group of instructions can be shortened to

```
DOA ACS , ACTR

DOBS ACD, ACTR
```

Once the datum has been sent, the transmitter subsection of the ACTR has no way of knowing whether the other computer has received it and strobed the data into the processor (i.e. it is ready to accept more data). It is therefore advisable to have the receiving computer send a data acknowledgement message using its own transmitter.

When a transmission is received, the Actr Done is set to 1. The method of ascertaining this condition will be dealt with in some length shortly but for now let us assume that we know that a reception has been completed. The A buffer of the receiver (data word) must be loaded into an accumulator as well as the B buffer (control bits). The Done flag also must be put to 0 by a clear pulse:

```
DIA  ACS, ACTR      ;READ DATA WORD INTO ACS

DIBC ACD, ACTR      ;READ CONTROL BITS INTO ACD
                    ;CLEAR ACTR DONE FLAG
```

Once the ACTR has completed the reception of a word (20 bits), it sets Done to 1. The program can determine this condition in one of two ways. By using the I/O SKP instruction, the program can test the condition of the Done flag.

```
SKPDN ACTR          ;IS DONE 1?

JMP .-1             ;NO...CONTINUE TO WAIT
                    ;YES...MESSAGE RECEIVED
```

Another way is to utilize the interrupt system that is standard on the NOVA (ECLIPSE) computer. The interrupt system is made up of an interrupt request line to which each I/O device is connected, an Interrupt On flag in the CPU, and a 16-bit interrupt priority mask. The Interrupt On flag controls the status of the interrupt system. If the flag is set to 1, the CPU will respond to and process interrupts. If the flag is set to Ø, the CPU will not respond to any interrupts. An interrupt request is initiated by an I/O device when it completes its operation. When the ACTR completes a message reception, as well as setting Done, it also places an interrupt request on the interrupt request line, provided that the bit in the interrupt priority mask (bit 8) which corresponds to the priority level of the device is Ø. If the mask bit is 1, the ACTR can set its Done to 1, but does not place an interrupt request on the interrupt request line.

If the Interrupt On flag is 1 at the time the processor completes execution of any instruction, the processor honours any requests on the interrupt request line. If the Interrupt

On flag is 0, the CPU does not look at the interrupt request line, it just continues to the next sequential instruction. The CPU honours an interrupt request by setting the Interrupt On flag to 0 so that no interrupts can interrupt the first part of the interrupt service routine. The CPU then places the updated program counter into physical memory location 0 and executes a "JMP @1" instruction. It is assumed that physical location 1 contains the address, either direct or indirect, of the interrupt service routine.

Once the CPU has transferred control to the interrupt service routine, it is up to that routine to save any accumulators that will be used, save the carry bit if it will be used, determine which device requested the interrupt, and then service the interrupt. The determination of which device needs service can be done by I/O SKP instructions or the routine can use the INTERRUPT ACKNOWLEDGE instruction.

The INTERRUPT ACKNOWLEDGE instruction (INTA) returns the 6-bit code of the device requesting the interrupt. If more than one device is requesting service, the code returned is the code of that device requesting an interrupt which is physically closest to the CPU on the I/O bus. After servicing the device, the interrupt routine should restore all saved values, set the Interrupt On flag to 1 and return to the interrupted program. The instruction that sets the Interrupt On flag to 1 (Interrupt Enable) allows the processor to execute one more instruction (if the Interrupt Enable instruction changed the condition of the Interrupt On flag) before the next interrupt can take place. In order to

prevent the interrupt service routine from locking itself into a loop, this next instruction should be the instruction that returns control to the interrupted program. Since the updated value of the program counter was placed in location Ø by the CPU before honouring the interrupt, all the interrupt routine has to do, after restoring the AC's and the carry bit, is to execute an INTERRUPT ENABLE instruction and a "JMP @ Ø" instruction and control will be returned to the interrupted program.

If the Interrupt On flag remains 0 throughout the interrupt service routine, the interrupt routine cannot be interrupted and there is only 1 level of device priority. This level is determined by either the order in which the I/O SKP instructions are issued or (if the INTERRUPT ACKNOWLEDGE is used) by the physical location of the devices on the bus. In systems of widely differing speeds, such as a teletypewriter versus a fixed head disc, the programmer may wish to set up a multiple level interrupt scheme. Hardware and instructions are available on the NOVA (ECLIPSE) computers to allow the implementation of priority interrupts.

Each of the I/O devices is connected to a bit in the 16 bit priority mask. Devices which operate at roughly the same speed are connected to the same bit in the mask. Even though standard mask bit assignments have higher number bits assigned to low speed devices, no implicit ordering is intended. The manner in which these priority levels are ordered is completely up to the programmer.

The condition of the priority mask is altered by the MASK OUT instruction. If a bit in the priority mask is set to 1,

then all devices in the priority level corresponding to that bit will be prevented from requesting on interrupt when they complete an operation. In addition all pending interrupt requests from devices in that priority level are disabled. The priority mask bit for the ACTR is bit 8.

If the receiving computer has many calculations to perform, it cannot handle an incoming ACTR transmission by continually testing the state of the Done. It is necessary then to use the interrupt system to allow the processor to do other computing between interrupts and only reference the ACTR when it has received a complete transmission. This scheme also allows the processor to recognize data transmission as soon as it is received rather than periodically checking the ACTR for a new message.

Use is made of the priority level interrupt scheme in operating systems such as Data General RDOS (Real Time Disc Operating System) and RTOS (Real Time Operating System). These systems supply a multi-priority level interrupt handler. Devices are included in this handler either at system generation or during run-time. When an interrupt is detected by the hardware, the currently executing program is suspended and control goes to an interrupt dispatch program. This routine directs control to the correct servicing routine by using the device code (obtained by an INTA instruction) as an index into an interrupt branch table. The entry in this table is the address of a device control table (DCT) associated with the device service routine. This table contains the service mask that is to be ORed with the current

interrupt mask while control is in the device service routine. This mask establishes whick devices if any will be allowed to interrupt the currently interrupting device. Since interrupts are enabled when control is passed to a device service routine, the interrupt service mask must contain the mask bit for the device being serviced.

The handling of the ACTR through the facilities of RDOS/RTOS relieves the programmer of the responsibilities of maintainig program continuity during interrupts and implementing a multi-level interrupt handler. RDOS/RTOS is very useful in handling two problems which arise as a result of design constraints of the ACTR.

IN a multi-tasking environment, there will be competition for system resources (the ACTR among many others). A task currently using the ACTR must be assured that no other routine will also attempt to initiate a data transfer. The RDOS/RTOS facility of sync words in a multi-tasking environment allows a routine to gain exclusive control of the ACTR hardware and software and maintain control until it has finished its transmission and any accompanying receptions. ACTR control would always be allocated on a priority basis and any requests for the ACTR, that are issued while it is in use, would be queued until the ACTR became free.

When using the ACTR, the transmitting computer must await an acknowledgement of reception before sending more data. If both systems in the ACTR begin to transmit simultaneously, data could easily be lost when the ACTR receiver routines mistake incoming data for an acknowledgement of the data it has sent.

The user must design the ACTR handlers to be able to accommodate an occurence of this kind or so design the system configuration to eliminate the possibility of the two CPU's attempting to transmit simultaneously.

In a master-slave environment, the competition for the ACTR hardware between systems can never occur. In this mode, the master computer would always initiate communication between the two processors. The slave system would only transmit to fulfill requests from the master. The task facilities of RDOS/RTOS allows one task to be dedicated to servicing the ACTR in the slave CPU. This task would gain CPU control only when the ACTR receiver generates an interrupt. Once the message reception is completed, this task would determine what further action has to be taken.

Appendix A of this report will deal with the implementation of a software handler for the ACTR in a multi-tasking environment for a master/slave system.

ACTR COMMAND

DIA	⟨f⟩ ac,	ACTR	read receiver control bits
DIB	⟨f⟩ ac,	ACTR	read receiver data word
DOA	⟨f⟩ ac,	ACTR	send data word to transmitter
DOB	⟨f⟩ ac,	ACTR	send control bits to transmitter
NIO	⟨f⟩	ACTR	no I/O transfer
SKP	⟨t⟩	ACTR	skip if ⟨t⟩ is true

t	BN	tests for Busy = 1
	BZ	tests for Busy = 0
	DN	tests for Done = 1
	DZ	tests for Done = 0
f	C	will clear Done and clear ACTR interrupt request
	S	will set Busy and begin transmission

CENTRAL PROCESSOR FUNCTIONS

INTA	ac	will return 4Ø to specified accumulator if the ACTR is the nearest device to the processor requesting an interrupt
MSKO	ac	will prevent the ACTR from generating an interrupt if bit 8 is set in the specified accumulator.
IORST		will clear the Done and Busy of the ACTR and the Interrupt Disable flag (equivalent of a MSKO with 8-= Ø). The Interrupt Request of the ACTR will also be cleared.

ACTR CONTROL SUMMARY

Figure 9

In summary, the control logic for the ACTR arises quite logically out of the DG I/O structure. A small number of I/o commands allow complete control of the ACTR and its link into the interrupt structure and the general control store of the central processor. Figure 9 contains a brief resume of the dedicated instructions that control the ACTR and the central processor functions that influence the ACTR as part of their general operation.

CHAPTER 4

CONCLUSIONS

The transfer of usable information between any computer systems is always a complex undertaking: the information must be put into a format that the other processor can accept and act on, the integrity of the transferred information must be ensured, synchronization of the systems must be accomplished so that data is not lost between the systems and data must be moved between the machines as quickly as possible.

The ACTR can fulfill the above requirements when used in the proper operating environment. The ACTR master/slave handler will transmit messages formatted by the user and perform parity checking on all received information. Since data is completely assembled in the ACTR receiver before it is read in and data acknowledgements are necessary, processor-processor synchronization is not a problem. The slower speed of the ACTR, when compared with the MCA is out-weighed in many applications by its increased range and error checking capabilities. The reliability of the ACTR hardware is such that its transfer rate can be safely increased by the elimination of all error checking. (The user should however, use the ACTR diagnostic in APPENDIX B to ensure that the hardware transfer clocks are not marginally adjusted.)

Minicomputer multiprocessor systems, such as those based on Data General NOVA and ECLIPSE processors, represent one of the most rapidly growing areas in computer development. Their

inherent versatility and sizable computer power make them an ideal medium for real-time instrumentation control and simulation systems. With the reliance these systems place on shared processor resources, data transfer devices such as the ACTR will continue to be an important system compound.

APPENDIX A

ACTR MASTER-SLAVE HANDLING ROUTINES

An important function of any real time operating system is the efficient handling of input-output operations. Optimum usage of machine devices and central processor time in the accomplishment of tasks is the real reason for designing and implementing a multi-tasking system.

The responsibility of RDOS/RTOS I/O control is to react during normal program execution to the structuring of I/O requests, making assignment of requests to machine devices when they are idle, and queuing requests for devices which are busy. Through the queuing facility, RDOS/RTOS makes it possible to achieve maximum and continuous overlap of multi-tasks without direct intervention by the tasks themselves. The handler for the ACTR in the master-slave environment conforms to these requirements, enabling tasks to access the ACTR with minimal pre-processing.

The handler for the ACTR is composed of three sections: a transmission routine (SENDM), a reception routine (RECM) and an identification and interrupt routine (IDACTR). These routines exist, with only minor differences, in both the master and slave processor.

The call to the routine SENDM is similar in structure to a RDOS/RTOS SYSTEM call as it incorporates both an error and normal return. Control of SENDM is allocated on a priority basis to calling tasks through use of the RDOS/RTOS facility of sync

ø

DESTINATION

WORD COUNT

MESSAGE #

(ACTUAL MESSAGE)

ACTR MASTER-SLAVE MESSAGE FORMAT

Figure 1ø

words. Once a task gains control of SENDM through a sync word, it maintains exclusive control until all data transmissions requested by that task are completed.

A calling task passed SENDM the address of the message to be transmitted. This message must be constructed in accord with the format illustrated in Figure IO. The actual message transmitted begins with the word count; the first two words are for the use of SENDM only and are not included in the word count. The control bits, which accompany each transmitted data word, are used to indicate the start of a transmission, end of transmission and the parity of the transmitted data. The control bits are generated by SENDM and are not passed beyond the ACTR routines.

After each data word is transmitted, the SENDM routine goes into task suspension awaiting the reception of the data acknowledgement from the receiving computer. Once the reception acknowledgement is received, it is checked for a data error message. If a data error is confirmed, the transmission is terminated, ACTR control is released and the error return is taken from SENDM.

If a task in the master CPU, that calls SENDM expects a return message from the slave CPU, the DESTINATION word in the message that it passes to SENDM must contain the address for the reception of the incoming data. If no return is expected, this word must be set to -1. When SENDM has completed transmitting its message, the DESTINATION is checked for a valid address. If it is found, control is passed to RECM, the reception handler, otherwise the control of SENDM is released to the system and the

normal return is taken back to the calling task.

The reception routine, RECM, is not directly accessible to the user. It is called by SENDM in the master CPU if a return message is expected from the slave CPU. In the slave CPU, RECM is part of the reception task (ACTR) which is responsible for all receptions and transmission in that processor.

RECM awaits each incoming data transmission by a REC to a sync word. This sync word is activated by an IXMT from the ACTR interrupt handler upon the reception of data into the receiver buffers. Once RECM is unsuspended, a parity check is performed on the incoming data. If a parity error is detected in the received words, an error message is returned to the transmitting computer in the data acknowledgement. If no error is detected, the received data is stored, and a successful reception is signalled.

In the master CPU, the core area for the reception of data from the slave is contained in the DESTINATION word of the message that initiated the response from the slave. Once the entire message is received, control of the ACTR is returned to the system and the normal return is taken from SENDM. In the event of a reception error, ACTR control is relinquished and the error return is taken.

The slave CPU uses RECM as part of the ACTR service task. The system initializer is responsible for tasking off this monitoring task (entry point ACTR). No other task in the slave CPU is allowed any access to the ACTR. The slave uses the received message number to determine the core area that is to be used to

store the incoming transmission. This message number also determines which subroutine is to be called to handle the message after it has been completely assembled in the core. After control is returned from this post-reception processing, the ACTR task awaits the next message.

The message handling routines called from RECM in the slave CPU may call SENDM to return a message to the master computer or introduce changes in the control functions of the slave CPU. The action taken by these called handlers is left completely up to the user.

For the ACTR handlers, RECM and SENDM, to operate properly, it is necessary that an interrupt handler for the ACTR be included in the interrupt dispatch table of RTOS/RDOS. This can be done either at SYSGEN or during the execution program. Any device that issues an interrupt that is not included in the dispatch table will have its interrupt cleared with a resulting loss of information.

The routine IDACTR links the ACTR interrupt handler (IACTR) into the RTOS/RDOS interrupt structure through use of a run-time system call. It then releases control of the SENDM sync word to the system. By locking out SENDM until the device identification is completed, the users of the ACTR are assured that no information will be lost during system initialization.

As was mentioned before, SENDM and RECM vary slightly between the master and slave computers. Through use of the capabilities of the DG MACRO Assembler, one source file can be made to include both the master and slave versions. The assembly of

the respective relocatable binary files is controlled by an equate switch (MASTER). When the switch MASTER = 1, the master CPU version is assembled and when MASTER = 0, the slave code is generated. This switch must be defined in an equate file which proceeds the ACTR handler source file (INTERLINK) in the MAC string. The version of the ACTR handler presented in this Appendix is that appropriate to the slave CPU. Those parts of the source file that applied only to the master computer are listed but have not generated any code.

```
01
02
03
04
05
06
07
08
09
10
11              ;*****************************************************
12              ;
13              ; 01 TITLE
                .TITL LINK  ; ACTR HANDLER
15              ;
16              ;*****************************************************
17              ;
18              ; 02 IDENTIFICATION
19              ;
20              ;  NAME:    INTERLINK.SR
21              ;
22              ;  PURPOSE:  THIS PROGRAM SUPPLIES TRANSMISSION AND
23              ;            RECEPTION ROUTINES FOR THE ACTR
24              ;
25              ;  AUTHOR:  MICHAEL BRETT
26              ;
27              ;  ORIGINAL ISSUE DATE:  1/2/77
28              ;
29              ;  ISSUE:  1
30              ;
31              ;  DATE OF ISSUE:  1/2/77
32              ;
33              ;*****************************************************
```

```
01
02
03
04
05
06              ;*****************************************************
07              ;
08              ; 03 DESCRIPTION
09              ;
10              ;  NARRATIVE:
11              ;
12              ;    THIS ROUTINE IS A SOFTWARE INTERFACE TO THE ACTR.
13              ;  IT OPERATES IN A MASTER/SLAVE ENVIROMENT WITH THE
14              ;  MASTER CPU INITIATING ALL DATA TRANSFERS.
15              ;                MESSAGE FORMAT IS
16              ;                        Ø
17              ;                    DESTINATION
18              ;                    WORD COUNT
19              ;                    MESSAGE #
20              ;                  (ACTUAL MESSAGE)
21              ;  WORD COUNT DOES NOT INCLUDE LEADING Ø OR
22              ;  DESTINATION. MESSAGE # CORRESPONDS TO A TABLE ENTRY
23              ;  IN SLAVE CPU, IGNORED IN RECEPTION IN MODEL.
24              ;  DESTINATION IN MODEL IS -1 IF NO RETURN MESSAGE
25              ;  IS EXPECTED; CONTAINS RECEPTION ADDRESS IF
26              ;  RETURN IS EXPECTED; IGNORED IN THE SLAVE.
27              ;  FIRST 2 WORDS OF MESSAGE ARE NOT TRANSMITTED.
28              ;     WHEN SENDM IS CALLED, AC2 MUST CONTAIN A
29              ;  POINTER TO THE START OF THE MESSAGE. AN STX IS
30              ;  SENT WITH THE WORD COUNT AND AN ETX WITH THE
31              ;  LAST WORD. THE RECEIVING COMPUTER WILL PERFORM
32              ;  A PARITY CHECK ON THE INCOMING DATA. IF AN
33              ;  ERROR IS DETECTED, IT WILL SEND AN ERROR MESSAGE
34              ;  IN THE DATA ACKNOWLEDGEMENT. IF SENDM RECEIVES
35              ;  AN ERROR MESSAGE NO FURTHER TRANSFERS WILL BE
36              ;  ATTEMPTED AND THE ERROR EXIT WILL BE TAKEN.
37              ;     IDACTR WILL IDEF THE ACTR TO RTOS/RDOS
38              ;  AND RELEASE THE SENDM SYNC WORD SO THAT SENDM
39              ;  CAN BE ACCESSED.
40              ;     THE SLAVE CPU RECEPTION TASK (.ACTR) WILL
41              ;  USE THE MESSAGE # TO FIND THE RECEPTION AREA
42              ;  FROM AN EXTERNAL TABLE (TABLE) AND TO FIND
43              ;  THE HANDLING ROUTINE FOR THE MESSAGE FROM
44              ;  ANOTHER EXTERNAL TABLE (HNDLR). .ACTR MUST
45              ;  BE TASKED OFF DURING SYSTEM INITIALIZATION.
46              ;*****************************************************
```

```
!0003 LINK
01
02
03
04
05
06
07              ;*************************************************************
08              ;
09              ; 04 CALLING SEQUENCE
10              ;
11              ;  ^^^^SENDM^^^^
12              ;       AC2: START OF MESSAGE HEADER
13              ;      CALL: JSR @.+1
14              ;            SENDM
15              ;            ERROR RETURN
16              ;            NORMAL RETURN
17              ;    RETURN: NONE
18              ;            ALL ACCUMULATORS AND CARRY
19              ;                  DESTROYED
20              ;
21              ;  ^^^^IDACTR^^^^
22              ;            NO ACCUMULATORS PASSED
23              ;      CALL: JSR @.+1
24              ;            IDACTR
25              ;    RETURN: NONE
26              ;            ALL ACCUMULATORS AND CARRY
27              ;                  DESTROYED
28              ;
29              ;  ^^^^.ACTR^^^^
30              ;            CREATED AS ACTR RECEPTION TASK
31              ;            IN SLAVE CPU
32              ;
33              ;*************************************************************
```

```
!0004 LINK
01
02
03
04                ;*********************************************************
05                ;
06                ;  05 SUBROUTINES
07                ;
08                ;          THE LINK STRING MUST INCLUDE A FILE WHICH
09                ;       DEFINES THE PARAMETER MASTER. MASTER=1
10                ;       FOR A MASTER CPU COMPILE AND MASTER=0
11                ;       FOR A SLAVE CPU COMPILE.
12                ;          THIS EQUATE MUST APPEAR FIRST IN MAC STRING
13                ;
14                ;*********************************************************
15                ;
16                ; 06 EXTERNAL DATA
17                ;
18                .EXTN .IXMT                ; RDOS/RTOS TASK CALL
19                .EXTN .XMT                 ; RDOS/RTOS TASK CALL
20                .EXTN .REC                 ; RDOS/RTOS TASK CALL
21                .EXTN .UIEX                ; RDOS/RTOS TASK CALL
22        000001  .IFE MASTER                ; SLAVE COMPUTER
23                .EXTD HNDLR                ; LOCATION OF HANDLER TABLE
24                .EXTD TABLE                ; LOCATION OF RECEPTION
25                .ENDC
26                ;
27                ;*********************************************************
28                ;
29                ; 07 PROGRAM
30                ;
31                ;
32        000040  .DUSR ACTR=40
33        006401  .DEUR EJSR=6401
34                ;
35                .ENT IDACTR
36                .ENT SENDM
37        000001  .IFE MASTER
38                .ENT .ACTR
39                .ENDC
40                ;
41                .NREL                      ; PAGE 1 RELOCATABLE
42
```

```
!0005 LINK
01
02
03                                  ; IDENTIFY ACTR TO RTOS/RDOS
04                                  ;       NO ACCUMULATORS PASSED
05                                  ; CALL: JSR @.+1
06                                  ;       IDACTR
07                                  ;RETURN:ALL ACCUMULATORS AND
08                                  ;       CARRY DESTROYED
09
10
11                IDACTR:
12 00000'054415         STA 3,.RTRN     ; SAVE RETURN ADDRESS
13 00001'020545         LDA 0,.40       ; DEVICE CODE OF ACTR
14 00002'024412         LDA 1,.AYDCT    ; ADDRESS OF ACTR DCT
15 00003'006017         .SYSTM          ; IDENTIFY ACTR TO SYSTEM
16 00004'021007         .IDEF
17 00005'063077         HALT            ; SYSTEM ERROR !!!
18 00006'020527         LDA 0,SSEND     ; ADDRESS OF SENDM SYNC
19 00007'126000         ADC 1,1         ; MAKE AC1 NON-ZERO
20 00010'077777         .XMT            ; NOW SAFE TO RELEASE SENDM
21 00011'063077         HALT            ; SYSTEM ERROR
22 00012'030403         LDA 2,.RTRN     ; GET RETURN ADDRESS
23 00013'001001         JMP 1,2         ; BUMP BY ONE FOR EXTENDED CALL
24                                      ; RETURN TO CALL
25
26 00014'000034'.AYDCT: AYDCT           ; ADDRESS OF ACTR DCT
27 00015'000000 .RTRN:  0               ; RETURN ADDRESS
28
29
30                                      ; ACTR INTERRUPT HANDLER
31 00016'050414 IACTR:  STA 2,AC2       ; SAVE AC2
32 00017'054414         STA 3,AC3       ; SAVE AC3
33 00020'064440         DIA 1,ACTR      ; GET CONTROL WORD
34 00021'044522         STA 1,RCONT     ; SAVE IT
35 00022'065640         DIBC 1,ACTR     ; GET DATA WORD
36                                      ; AND CLEAR INTERRUPT
37 00023'044517         STA 1,RDATA     ; SAVE DATA WORD
38 00024'126000         ADC 1,1         ; MAKE AC1 NON-ZERO
39 00025'077777         .IXMT           ; SIGNAL MESSAGE RECEIVED
40 00026'063077         HALT            ; SYSTEM ERROR !!!
41 00027'030403         LDA 2,AC2       ; RESTORE AC2
42 00030'034403         LDA 3,AC3       ; RESTORE AC3
43 00031'077777         .UIEX           ; RETURN CONTROL TO SYSTEM
44
45 00032'000000 AC2:    0               ; AC2 STORAGE
46 00033'000000 AC3:    0               ; AC3 STORAGE
47
48 00034'000037'AYDCT:  AREA            ; COMPATIBILTITY WITH RTOS
49 00035'000377         377             ; INTERRUPT SERVICE MASK
50 00036'000016'        IACTR           ; INTERRUPT SERVICE POINTER
51 00037'000010 AREA:   .BLK 8.         ; RTOS SAVE AREA
52
```

```
!0006 LINK
01                                      ; ACTR TRANSMISSION ROUTINE
02                                      ;       AC2: MESSAGE ADDRESS
03                                      ;CALL:  JSR @.+1
04                                      ;       SENDM
05                                      ;       (ERROR RETURN)
06                                      ;       (NORMAL RETURN)
07                                      ;RETURN:NO MESSAGES RETURNED
08                                      ;       ALL ACCUMULATORS AND CARRY
09                                      ;       DESTROYED
10
11                      SENDM:
12 00047'055000                 STA 3,0,2       ; SAVE RETURN ADDRESS IN HEADER
13 00050'020465                 LDA 0,SSEND     ; SENDM SYNC ADDRESS
14 00051'077777                 .REC            ; CAPTURE ACTR CONTROL
15 00052'050460                 STA 2,HEAD      ; SAVE ADDRESS OF MESSAGE
16 00053'151400                 INC 2,2         ;
17 00054'151400                 INC 2,2         ; POINT AC2 PAST HEADER WORDS
18 00055'050464                 STA 2,DATPTR    ; STORE IN TRANSMISSION POINTER
19 00056'025000                 LDA 1,0,2       ; GET WORD COUNT
20 00057'044466                 STA 1,WC        ; SAVE IT
21 00060'014465                 DSZ WC          ; REALLY NEED WC-1
22 00061'030456                 LDA 2,STX       ; GET START CONTROL BITS
23
24 00062'034457 LOOP:           LDA 3,DATPTR    ; GET DATA ADDRESS
25 00063'035400                 LDA 3,0,3       ; GET ACTUAL DATA
26 00064'076040                 DOB 3,ACTR      ; PUT IN TRANSMITTER BUFFER
27                                              ; NOW DETERMINE PARITY
28 00065'102400                 SUB 0,0
29 00066'151004                 MOV 2,2,SZR     ; IS CONTROL WORD NON-ZERO ?
30 00067'101400                 INC 0,0         ; YES... INC PARITY COUNTER
31                                              ; NOW DO DATA WORD
32 00070'175122                 MOVZL 3,3,SZC   ; IS SHIFTED BIT SET ?
33 00071'101400                 INC 0,0         ; YES.. INC PARITY COUNTER
34 00072'175004                 MOV 3,3,SZR     ; FINISHED DATA WORD ?
35 00073'000775                 JMP .-3         ; NO.. CONTINUE IN LOOP
36 00074'024453                 LDA 1,.14       ; GET PARITY ON BITS
37 00075'101222                 MOVZR 0,0,SZC   ; ODD # OF BITS SET ?
38 00076'133000                 ADD 1,2         ; YES... PUT ODD PARITY BITS
39                                              ; INTO CONTROL WORD
40 00077'071140                 DOAS 2,ACTR     ; PUT IN TRANSMITTER BUFFER
41                                              ; START TRANSMISSION
42 00100'020433                 LDA 0,TRANSM    ; WAIT FOR ACKNOWLEDGEMENT
43 00101'000051'                .REC
44 00102'024441                 LDA 1,RCONT     ; GET CONTROL BITS
45 00103'125004                 MOV 1,1,SZR     ; IS IT NON-ZERO ?
46 00104'000422                 JMP FLOP        ; YES... TRANSMISSION ERROR
47 00105'024433                 LDA 1,ETX       ; GET END OF TRANSMISION BITS
48 00106'147414                 AND# 2,1,SZR    ; WAS END OF MESSAGE SENT ?
49       000000         .IFN MASTER             ; MASTER CPU
50                              JMP IFBACK      ; YES.. CHECK IF MESSAGE
51                                              ; IS TO RETURNED
52                      .ENDC
53       000001         .IFE MASTER             ; SLAVE CPU
54 00107'000407                 JMP NEXIT       ; TAKE NORMAL EXIT
55                                              ; NO RETURN MESSAGES
56                                              ; IN SLAVE CPU
57                      .ENDC
```

```
!0007 LINK
01                                          ; DIDN'T SEND LAST WORD
02
03 00110'152400          SUB 2,2            ; NORMAL CONTROL BITS
04 00111'014434          DSZ WC             ; DECREMENT WORD COUNT
05 00112'000402          JMP .+2            ; NOT AT ZERO
06 00113'030425          LDA 2,ETX          ; AT ZERO.. GET ETX BITS
07 00114'010425          ISZ DATPTR         ; INCREMENT STORAGE POINTER
08 00115'000745          JMP LOOP           ; SEND NEXT WORD
09
10
11
12
13       000000 .IFN MASTER                 ; MASTER CPU
14                                          ; MESSAGE RETURN CHECK
15              IFBACK: LDA 2,HEAD          ; GET START OF MESSAGE
16                      LDA 2,1,2           ; GET DESTINATION ADDRESS
17                      COM# 2,2SNR         ; IS IT -1 ?
18                      JMP NEXIT           ; YES.. NO RETURN EXPECTED
19                      JMP RECM            ; NO.. RECEIVE MESSAGE
20                                          ; AC2=RECEPTION AREA
21              .ENDC
22
23
24
25
26
27                                          ; SETUP FOR NORMAL RETURN
28 00116'024426 NEXIT:  LDA 1,.3            ; AC3=3
29 00117'032413          LDA 2,@HEAD        ; GET RETURN ADDRESS
30 00120'133000          ADD 1,2            ; BUMP BY 3 FOR NORMAL RETURN
31
32                                          ; SENDM EXIT
33 00121'020414 EXIT:    LDA 0,SSEND        ; GET ADDRESS OF SENDM SYNC
34 00122'126000          ADC 1,1            ; MAKE AC1 NON-ZERO
35 00123'000010'         .XMT               ; RELEASE SENDM CONTROL
36 00124'063077          HALT               ; SYSTEM ERROR !!!
37 00125'001000          JMP 0,2            ; RETURN TO CALL
38
39
40                                          ; SETUP FOR ERROR EXIT
41 00126'032404 FLOP:    LDA 2,@HEAD        ; GET MESSAGE ADDRESS
42 00127'151400          INC 2,2            ; BUMP BY 2 FOR ERROR RETURN
43 00130'151400          INC 2,2            ;
44 00131'000770          JMP EXIT           ; LEAVE SENDM
45
46
```

```
01
02
03
04
05
06                          ; CONTROL WORDS FOR ACTR ROUTINES
07
08
09
10 00132'000000 HEAD:    0                  ; START OF MESSAGE
11 00133'000134'TRANS:   .+1                ; TRANSMISSION COMPLETED SYNC
12 00134'000000          0
13 00135'000136'SSEND:   .+1                ; SENDM CONTROL SYNC
14 00136'000000          0
15
16 00137'000001 STX:     1                  ; STX CONTROL BIT
17 00140'000002 ETX:     2                  ; ETX CONTROL BIT
18 00141'000000 DATPTR:  0                  ; MESSAGE POINTER
19
20 00142'000000 RDATA:   0                  ; RECEIVED DATA WORD
21 00143'000000 RCONT:   0                  ; RECEIVED CONTROL WORD
22
23 00144'000003 .3:      3
24 00145'000000 WC:      0                  ; WORD COUNT
25
26 00146'000040 .40:     ACTR               ; DEV CODE OF ACTR
27 00147'000014 .14:     14                 ; PARITY INDICATORS
28
29
```

```
!0009 LINK
01
02
03
04
05          000001 .IFE MASTER                  ; SLAVE RECEIVER TASK
06 00150'006401 .ACTR:  EJSR IDACTR             ; IDENTIFY ACTR TO SYSTEM
07          000000'
08                                              ; NOW AWAIT RECEPTION IN
09                                              ; ROUTINE RECM
10                  .ENDC
11
12
13
14                                              ; RECEPTION ROUTINE
15 00152'020761 RECM:   LDA 0,TRANS             ; RECEIVER WAIT SYNC
16 00153'000101'        .REC                    ; WAIT FOR RECEPTION
17 00154'004451         JSR PARITY              ; DO PARITY CHECK
18 00155'020762         LDA 0,STX               ; NOW CHECK FOR STX
19 00156'024765         LDA 1,RCONT             ; GET CONTROL BITS
20 00157'107415         AND# 0,1,SNR            ; WAS STX SENT ?
21 00160'000422         JMP AHEAD               ; NO.. JMP AROUND
22          000000 .IFN MASTER                  ; MASTER CPU
23                      LDA 1,RDATA             ; GET WC
24                                              ; WC=FIRST WORD RECEIVED
25                      STA 1,WC                ; STORE IT IN WC
26                  .ENDC
27          000001 .IFE MASTER                  ; SLAVE CPU
28                                              ; GET RECEPTION AREA
29                                              ; FOR SLAVE
30 00161'030761         LDA 2,RDATA             ; RDATA CONTAINS WC
31 00162'176120         ADCZL 3,3               ; AC3=-2
32 00163'173000         ADD 3,2                 ; SUBTRACT 2 FROM WORD COUNT
33 00164'050761         STA 2,WC                ; TAKES CARE OF WC AND
34                                              ; MESSAGE #
35
36 00165'020746         LDA 0,TRANS             ; GET RECEPTION SYNC
37 00166'000153'        .REC                    ; WAIT FOR RECEPTION
38 00167'004436         JSR PARITY              ; CHECK PARITY
39
40 00170'034752         LDA 3,RDATA             ; GET MESSAGE #
41 00171'030002$        LDA 2,TABLE             ; START OF RECEPTION TABLE
42 00172'173000         ADD 3,2                 ; GET RECEPTION AREA POINTER
43 00173'031000         LDA 2,0,2               ; GET RECEPTION AREA
44
45 00174'024001$        LDA 1,HNDLR             ; GET START OF TABLE
46 00175'137000         ADD 1,3
47 00176'035400         LDA 3,0,3               ; GET HANDLER ADDRESS
48 00177'054402         STA 3,.HNDLR            ; STORE IN POINTER
49 00200'000752         JMP RECM                ; RECEIVE NEXT WORD
50
51 00201'000000 .HNDLR: 0                       ; HANDLING ROUTINE POINTER
52                  .ENDC
```

```
!0010 LINK
01
02
03 00202'034740 AHEAD:   LDA 3,RDATA      ; GET RECEIVED DATA
04 00203'055000          STA 3,0,2         ; STORE IT
05 00204'014741          DSZ WC            ; DECREMENT WC
06 00205'000402          JMP .+2           ; NOT AT 0... CONTINUE
07 00206'000405          JMP FINISH        ; CHECK FOR ETX
08 00207'102400          SUB 0,0           ; AC0=0
09 00210'061140          DOAS 0,ACTR       ; SEND DATA OK MESSAGE
10 00211'151400          INC 2,2           ; INCREMENT POINTER
11 00212'000740          JMP RECM          ; GET NEXT RECEPTION
12
13
14
15 00213'020725 FINISH:  LDA 0,ETX         ; GET ETX BITS
16 00214'107415          AND# 0,1,SNR      ; WAS ETX RECEIVED
17 00215'000405          JMP ERROR         ; NO... ERROR
18 00216'102400          SUB 0,0           ; YES.. PUT AC0=0
19 00217'061140          DOAS 0,ACTR       ; SEND DATA OK MESSAGE
20       000000 .IFN MASTER                ; MASTER CPU
21                       JMP NEXIT         ; COMPLETED RECM
22                                         ; TAKE NORMAL EXIT
23               .ENDC
24       000001 .IFE MASTER                ; SLAVE CPU
25 00220'006761          JSR @.HNDLR       ; JUMP TO ROUTINE
26 00221'000731          JMP RECM          ; AWAIT NEXT MESSAGE
27                                         ; TO HANDLE RECEPTION
28               .ENDC
29
30               ERROR:                    ; ERROR EXIT FROM
31                                         ; RECM
32 00222'126000          ADC 1,1           ; AC1=-1
33 00223'065140          DOAS 1,ACTR       ; SEND DATA ERROR
34                                         ; MESSAGE
35       000000 .IFN MASTER                ; MASTER CPU
36                       JMP FLOP          ; TAKE ERROR EXIT
37                                         ; FROM SENDM
38               .ENDC
39       000001 .IFE MASTER                ; SLAVE CPU
40 00224'000726          JMP RECM          ; AWAIT NEXT MESSAGE
41               .ENDC
```

```
01
02
03                                          ; RECEPTION PARITY CHECK
04 00225'102400 PARITY: SUB 0,0             ; ZERO PARITY COUNTER
05 00226'024715         LDA 1,RCONT         ; GET CONTROL BITS
06 00227'125122         MOVZL 1,1,SZC       ; BIT SET ?
07 00230'101400         INC 0,0             ; YES.. INC PARITY COUNTER
08 00231'125004         MOV 1,1,SZR         ; FINISHED WORD ?
09 00232'000775         JMP .-3             ; NO... CONTINUE IN LOOP
10 00233'024707         LDA 1,RDATA         ; GET DATA WORD
11 00234'125122         MOVZL 1,1,SZC       ; BIT SET IN DATA WORD ?
12 00235'101400         INC 0,0             ; YES.. INC PARITY COUNTER
13 00236'125004         MOV 1,1,SZR         ; FINISHED DATA WORD ?
14 00237'000775         JMP .-3             ; NO.. CONTINUE IN LOOP
15                                          ; PARITY COUNT LOOP
16                                          ; COMPLETED
17 00240'024703         LDA 1,RCONT         ; GET CONTROL BITS
18 00241'101212         MOVR# 0,0,SZC       ; EVEN # OF BITS SET
19 00242'000405         JMP ODD             ; NO... CHECK THAT ODD PARITY
20                                          ; BITS ARE SET
21                                          ; FOR EVEN NONE ARE SET
22 00243'020704         LDA 0,.14           ; GET PARITY MASK
23 00244'107405         AND 0,1,SNR         ; ARE PARITY BITS SET ?
24 00245'001400         JMP 0,3             ; NO.. TAKE NORMAL RETURN
25 00246'000754         JMP ERROR           ; YES.. TAKE ERROR EXIT
26
27 00247'020700 ODD:    LDA 0,.14           ; GET PARITY ASK
28 00250'107400         AND 0,1             ; AND CONTROL TO PARITY
29 00251'106405         SUB 0,1,SNR         ; ARE BOTH BITS SET ?
30 00252'001400         JMP 0,3             ; YES.. TAKE NORMAL RETURN
31 00253'000747         JMP ERROR           ; NO.. TAKE ERROR EXIT
32
33
34
35                      .END

**00000 TOTAL ERRORS, 00000 PASS 1 ERRORS
```

```
AC2     000032'        5/31    5/41    5/45
AC3     000033'        5/32    5/42    5/46
AHEAD   000202'        9/21   10/03
AREA    000037'        5/48    5/51
AYDCT   000034'        5/26    5/48
DATPT   000141'        6/18    6/24    7/07    8/18
ERROR   000222'       10/17   10/30   11/25   11/31
ETX     000140'        6/47    7/06    8/17   10/15
EXIT    000121'        7/33    7/44
FINIS   000213'       10/07   10/15
FLOP    000126'        6/46    7/41   10/36
HEAD    000132'        6/15    7/15    7/29    7/41    8/10
HNDLR   000001$ XD     4/23    9/45
IACTR   000016'        5/31    5/50
IDACT   000000' EN     4/35    5/11    9/06
LOOP    000062'        6/24    7/08
MASTE   000000         4/22    4/37    6/49    6/53    7/13    9/05    9/2
                       9/27   10/20   10/24   10/35   10/39
NEXIT   000116'        6/54    7/18    7/28   10/21
ODD     000247'       11/19   11/27
PARIT   000225'        9/17    9/38   11/04
RCONT   000143'        5/34    6/44    8/21    9/19   11/05   11/17
RDATA   000142'        5/37    8/20    9/23    9/30    9/40   10/03   11/1
RECM    000152'        7/19    9/15    9/49   10/11   10/26   10/40
SENDM   000047' EN     4/36    6/11
SSEND   000135'        5/18    6/13    7/33    8/13
STX     000137'        6/22    8/16    9/18
TABLE   000002$ XD     4/24    9/41
TRANS   000133'        6/42    8/11    9/15    9/36
WC      000145'        6/20    6/21    7/04    8/24    9/25    9/33   10/(
.14     000147'        6/36    8/27   11/22   11/27
.3      000144'        7/28    8/23
.40     000146'        5/13    8/26
.ACTR   000150' EN     4/38    9/06
.AYDC   000014'        5/14    5/26
.HNDL   000201'        9/48    9/51   10/25
.IXMT   000025' XN     4/18    5/39
.REC    000166' XN     4/20    6/14    6/43    9/16    9/37
.RTRN   000015'        5/12    5/22    5/27
.UIEX   000031' XN     4/21    5/43
.XMT    000123' XN     4/19    5/20    7/35
```

APPENDIX B

ACTR DIAGNOSTICS

```
0001 ACTR  MACRO REV 04.00                10:49:23 01/08/77
                ;*****************************************************
                ;
                ;
                ; NAME: ACTR.SR   ASYNCHROUS COMUNICATION TRANSMITTER
                ;                        RECEIVER DIAGNOSTICS
                ;
                ; AUTHOR: MICHAEL BRETT
                ;
                ; REVISION HISTORY:
                ;    REV      DATE
                ;     00    AUG 1/76
                ;
                ; NARRATIVE:
                ; THIS PROGRAM WILL FIRST DISABLE RDOS. THE FLAGS AND
                ; INTERRUPTS OF THE ACTR WILL THEN BE TESTED. NEXT
                ; TRANSMISSION TESTS WILL BE CARRIED OUT. APPROPRIATE
                ; ERROR MESSAGES WILL BE OUTPUTTED. EXECUTION OF THE
                ; PROGRAM WILL BE TERMINATED WHEN AN INTERRUPT OR
                ; FLAG ERROR IS DETECTED. THE ERROR OUTPUT FROM THE
                ; THE TRANSMISSION PHASE CAN BE TERMINATED BY TYPING
                ; CONTROL C. ERROR STATISTICS WILL STILL BE OUTPUTTED.
                ; AT THE TEST TERMINATION, CONTROL IS RETURNED TO RDOS.
                ;
                ; OPERATION:
                ; CONNECT RECEIVER INPUT TO TRANSMITTER OUTPUT
                ; AND START PROGRAM
                ; THIS IS A STAND ALONE TEST OF A SINGLE ACTR BOARD
                ; AND REQUIRES USE OF THE RTC.
                ;
                ; NOTE:
                ; THIS PROGRAM CAN BE USED TO TEST THE INTERPROCESSOR
                ; LINK BY RUNNING THIS PROGRAM IN THE MASTER CPU AND
                ; USING THE SLAVE CPU TO REFLECT THE DATA.
                ;
                ;*****************************************************
```

```
!0002 ACTR
01
                        .TITL ACTR          ; ACTR DIAGNOSTIC
03
04                      .ENT START          ; START OF DIAGNOSTIC
05
06        000401        .COMM TASK,1*400+1
07
08                      .NREL
09
10        000040        .DUSR ACTR=40
11
12
13
14 00000'006017 START:  .SYSTM
15 00001'021000         .GHRZ               ; GET SYSTEM RTC FREQUENCY
16 00002'000401         JMP .+1
17 00003'101400         INC 0,0             ; INCREMENT IT BY 1
18 00004'040034-        STA 0,TIME          ; STORE IT
19 00005'060277         INTDS               ; DISABLE INTERRUPT
20 00006'020001         LDA 0,1             ; GET RDOS INTERRUPT POINTER
21 00007'040035-        STA 0,SAVE          ; SAVE IT
22 00010'020022-        LDA 0,INTPTR        ; GET ADDRESS OF SERV. ROUTINE
23 00011'040001         STA 0,1             ; STORE IT IN LOCATION 1
24 00012'102400         SUB 0,0
25 00013'062077         DOB 0,CPU           ; CLEAR INT MASK
26 00014'006000-        JSR @RITER
27 00015'000446'        MESSA               ; WRITE TITLES
28 00016'063511         SKPBZ TTO           ; WAIT FOR TTO TO FINISH
29 00017'000777         JMP .-1
30 00020'062677         IORST               ; CLEAR ALL FLAGS
31 00021'063440         SKPBN ACTR          ; SKIP IF BUSY=1
32 00022'000404         JMP .+4             ; OK FLAG 0
33 00023'006000-        JSR @RITER          ; ERROR!!!!!
34 00024'000520'        MES1                ; BUSY SHOULD EQUAL 0
35 00025'002001-        JMP @END            ; TERMINAL ERROR
36 00026'063640         SKPDN ACTR          ; SKIP IF DONE = 1
37 00027'000404         JMP .+4             ; OK FLAG 0
38 00030'006000-        JSR @RITER          ; ERROR!!!!!
39 00031'000544'        MES2                ; DONE SHOULD EQUAL 0
40 00032'002001-        JMP @END            ; TERMINAL ERROR
41 00033'102400         SUB 0,0             ; SET UP INTERRUPT FLAGS
42 00034'040005-        STA 0,TAG           ; TAG = 0
43 00035'101400         INC 0,0
44 00036'040002-        STA 0,TAG1          ; TAG1=1
45 00037'040003-        STA 0,TAG2          ; TAG2=1
46 00040'040004-        STA 0,TAG3          ; TAG3=1
47 00041'061014         DOA 0,RTC           ; SET CLOCK FREQUENCY TO 10HZ
48
49 00042'060177         INTEN               ; ENABLE INTERRUPT
50 00043'060114         NIOS RTC            ; START CLOCK
51 00044'000401         JMP .+1             ; PERFORM NO-OP TO
52 00045'000777         JMP .-1             ; WAIT FOR INTERRUPT
53
54                                          ; PREVIOUS SECTION USED TO
55                                          ; STABILIZE RTC FOR
56                                          ; ACTR INTERRUPT TESTS
57 00046'126440 CLOCK:  SUBO 1,1
58 00047'044002-        STA 1,TAG1          ; TAG1=0
59 00050'125400         INC 1,1
60 00051'044005-        STA 1,TAG           ; TAG=1
```

```
 0003 ACTR
01 00052'044003-          STA 1,TAG2        ; TAG2=1
02 00053'044004-          STA 1,TAG3        ; TAG3=1
03 00054'060114           NIOS RTC          ; START RTC
04                                          ; RTC STABILIZED AT 10 HZ
05 00055'060140           NIOS ACTR         ; START TRANSMITTER
06 00056'060177           INTEN             ; ENABLE INTERRUPT
07 00057'000401           JMP .+1           ; WAIT FOR INTERRUPT
08 00060'000777           JMP .-1
09
10
11                                          ; CLEAR FLAG TEST
12 00061'063740 BACK:     SKPDZ ACTR        ; SKIP IF DONE=0
13 00062'000404           JMP .+4           ; OK DONE IS 1
14 00063'006000-          JSR @RITER        ; ERROR
15 00064'000572'          MES21             ; PRINT ERROR MESSAGE
16 00065'002001-          JMP @END
17 00066'060240           NIOC ACTR         ; CLEAR DONE AND BUSY
18 00067'063440           SKPBN ACTR        ; IS BUSY 0
19 00070'000404           JMP .+4           ; YES OK
20 00071'006000-          JSR @RITER        ; NO , ERROR!!!!!
21 00072'000616'          MES22             ; WRITE ERROR MESSAGE
22 00073'002001-          JMP @END          ; TERMINAL ERROR
23 00074'063640           SKPDN ACTR        ; IS DONE 0
24 00075'000404           JMP .+4           ; YES OK
25 00076'006000-          JSR @RITER        ; NO, ERROR!!!!!
26 00077'000642'          MES23             ; WRITE ERROR MESSAGE
27 00100'002001-          JMP @END          ; TERMINAL ERROR
28 00101'126400           SUB 1,1
29 00102'044003-          STA 1,TAG2        ; TAG2=0
30 00103'125400           INC 1,1
31 00104'044005-          STA 1,TAG         ; TAG=1
32 00105'044002-          STA 1,TAG1        ; TAG1=1
33 00106'044004-          STA 1,TAG3        ; TAG3=1
34 00107'060114           NIOS RTC          ; START RTC
35 00110'060177           NIOS CPU          ; ENABLE INTERRUPT
36 00111'000401           JMP .+1
37 00112'000777           JMP .-1           ; WAIT FOR INTERRUPT
38
39
40
41                                          ; PRECEDING
42                                          ; TESTS IF NIOC WILL CLEAR
43                                          ; INTERRUPT REQUEST OF ACTR
44
45
46
47                                          ; TEST FOR ACTR MASK OUT
48 00113'024006-MASK:     LDA 1,.200        ; ACTR MASK
49 00114'066077           MSKO 1            ; MASK OUT ACTR INTERRUPT
50 00115'126440           SUBO 1,1
51 00116'044004-          STA 1,TAG3        ; TAG3=0
52 00117'125400           INC 1,1
53 00120'044002-          STA 1,TAG1        ; TAG1=1
54 00121'044005-          STA 1,TAG         ; TAG=1
55 00122'044003-          STA 1,TAG2        ; TAG2=1
56 00123'060114           NIOS RTC          ; START RTC
57 00124'060140           NIOS ACTR         ; START TRANSMITTER
58 00125'060177           INTEN             ; ENABLE INTERRUPT
59 00126'000401           JMP .+1           ; NO-OP FOR
60 00127'000777           JMP .-1           ; INTERRUPT
```

```
 0004 ACTR
01
02
03                                          ; IF EXECUTION REACHES HERE
04                                          ; FLAG TESTS COMPLETED
05                                          ; ALL FLAGS AND INTERRUPTS ARE
06                                          ; OPERATIONAL
07                                          ; NOW PROCEEDING WITH
08                                          ; TRANSMISSION TESTS
09 00130'006000-CONT:   JSR @RITER          ; WRITE TEST DESCRIPTIONS
10 00131'001220'        MES31
11 00132'020027-        LDA 0,NO            ; NO. OF WORDS TO BE TRANSMITTED
12 00133'040012-        STA 0,STRPTR        ; LOOP COUNTER FOR TRANSMISSION
13 00134'102400         SUB 0,0
14 00135'040007-        STA 0,CT1           ; ZERO ERROR COUNTER #1
15 00136'040010-        STA 0,CT2           ; ZERO ERROR COUNTER #2
16 00137'040013-        STA 0,STOP          ; STOP IS FOR PRINT SUPPRESS
17
18 00140'022012-LOOP:   LDA 0,@STRPTR       ; LOAD DATA WORD
19 00141'014012-        DSZ STRPTR          ; LOOP COMPLETED ?
20 00142'000402         JMP .+2             ; NO
21 00143'000445         JMP  FINISH         ; YES, WRITE OUT TRANS. STATS
22 00144'026012-        LDA 1,@STRPTR       ; LOAD CONTROL WORD
23 00145'014012-        DSZ STRPTR          ; LOOP COMPLETED ?
24 00146'000402         JMP .+2             ; NO
25 00147'000441         JMP FINISH          ; YES, WRITE OUT TRANS. STATS.
26 00150'030014-        LDA 2,.17           ; 17 IS A 4 BIT MASK
27 00151'147400         AND 2,1             ; CONTROL WORD IS 4 BITS LONG
28 00152'061240         DOAC 0,ACTR         ; OUTPUT DATA WORD TO BUFFER
29 00153'066140         DOBS 1,ACTR         ; OUTPUT CONTROL WORD TO BUFFER
30 00154'063640         SKPDN ACTR          ; HAS DATA BEEN RECEIVED
31 00155'000777         JMP .-1             ; NO
32 00156'070440         DIA 2,ACTR          ; READ DATA WORD FROM RECEIVER
33 00157'075640         DIBC 3,ACTR         ; READ CONTROL BITS FROM
34                                          ; RECEIVER
35 00160'040015-        STA 0,TEM0          ; SAVE
36 00161'044016-        STA 1,TEM1          ; ALL
37 00162'050017-        STA 2,TEM2          ; THE
38 00163'054020-        STA 3,TEM3          ; ACCUMULATORS
39 00164'142415         SUB# 2,0,SNR        ; DOES REC. = TRANS. DATA
40 00165'000403         JMP .+3             ; YES
41 00166'004541         JSR TRANS           ; NO, ERROR!!!!!!
42 00167'010007-        ISZ CT1             ; INCREMENT ERROR COUNTER
43 00170'024016-        LDA 1,TEM1          ;
44 00171'044015-        STA 1,TEM0          ; TEM0=TEM1
45 00172'034020-        LDA 3,TEM3
46 00173'054017-        STA 3,TEM2          ; TEM2=TEM3
47 00174'136415         SUB# 1,3,SNR        ; DOES REC. = TRAN. CONTROL BITS
48 00175'000743         JMP LOOP            ; YES
49                                          ; NO ERROR !!!!!!!
50 00176'010010-        ISZ CT2             ; INCREMENT ERROR COUNTER
51 00177'014013-        DSZ STOP            ; STOP=1 FOR PRINT SUPPRESS
52 00200'000402         JMP .+2
53 00201'000403         JMP .+3
54 00202'024021-        LDA 1,STAR          ; GET ASCII FOR STAR
55 00203'004462         JSR OUTPT           ; OUTPUT IT
56 00204'010013-        ISZ STOP            ; RESTORE STOP
57 00205'000401         JMP .+1
58 00206'004521         JSR TRANS           ; OUTPUT ERROR INFORMATION
59 00207'000731         JMP LOOP
60 00210'006000-FINISH: JSR @RITER          ; WRITE INFORMATION
```

```
 0005 ACTR
01 00211'001037'          MES40          ; ON # OF INCORRECT CONTROL
02 00212'024010-          LDA 1,CT2      ; WORDS TRANSFERRED
03 00213'004456           JSR CODE
04 00214'006000-          JSR@ RITER     ; WRITE INFORMATION ON #
05 00215'001054'          MES41          ; OF INCORRECT DATA
06 00216'024007-          LDA 1,CT1      ; WORDS TRANSFERRED
07 00217'004452           JSR CODE
08 00220'006000-          JSR @RITER
09 00221'001101'          MES42
10 00222'006000-ENDUP:    JSR@ RITER     ; TERMINATION OF PROGRAM
11 00223'001125'          MES43          ; WRITE MESSAGE
12 00224'063511           SKPBZ TTO      ; WAIT FOR TTO TO FINISH
13 00225'000777           JMP .-1
14 00226'024035-          LDA 1,SAVE     ; GET ADDRESS OF RDOS INT. HAND.
15 00227'044001           STA 1,1        ; PUT IT IN LOCATION 1
16 00230'062677           IORST          ; CLEAR ALL FLAGS
17 00231'014034-          DSZ TIME       ; IS THERE A  SYSTEM CLOCK
18 00232'000402           JMP .+2        ; YES
19 00233'000406           JMP AHEAD      ; NO
20 00234'020037-          LDA 0,.4       ; AC0=4
21 00235'024034-          LDA 1,TIME     ; SYSTEM FREQUENCY
22 00236'106415           SUB# 0,1,SNR   ; IS IT 4
23                                       ; 4 = 60 HZ
24 00237'126400           SUB 1,1        ; AC1=0 FOR
25                                       ; SETTING CLOCK TO 60 HZ
26 00240'065114           DOAS 1,RTC     ; SET FREQ AND START CLOCK
27 00241'060177 AHEAD:    INTEN          ; ENABLE INTERRUPT
28 00242'006017           .SYSTM
29 00243'004400           .RTN           ; RETURN CONTROL TO CLI
30
31
32                                       ; WRITE MESSAGE ROUTINE
33
34 00244'054417 WRITE:    STA 3,RETURN   ; LOCATION OF MESS. POINTER
35 00245'026416           LDA 1,@RETURN  ; LOCATION OF MESSAGE
36 00246'044416           STA 1,.PT      ; STORE IN AC1
37 00247'010414           ISZ RETURN     ; INCREMENT RETURN  ADDRESS
38 00250'020011-          LDA 0,.377
39 00251'026413 ERR1:     LDA 1,@.PT
40 00252'107415           AND# 0,1,SNR   ; CHECK FOR 0
41 00253'002410           JMP @RETURN    ; 0 IS THE TERMINATOR
42 00254'004411           JSR OUTPT      ; OUTPUT THE CHARACTER
43 00255'125300           MOVS 1,1
44 00256'107415           AND# 0,1,SNR   ; CHECK FOR 0
45 00257'002404           JMP @RETURN
46 00260'004405           JSR OUTPT      ; OUTPUT CHARACTER
47 00261'010403           ISZ .PT        ; INCREMENT POINTER
48 00262'000767           JMP ERR1       ; FETCH 2 MORE CHARACTERS
49 00263'000000 RETURN:   0
50 00264'000000 .PT:      0
51
52 00265'063511 OUTPT:    SKPBZ TTO      ; SKIP IF TTO NOT BUSY
53 00266'000777           JMP .-1
54 00267'065111           DOAS 1,TTO     ; OUTPUT AC1 TO TELETYPE
55 00270'001400           JMP 0,3        ; RETURN TO MAIN PROGRAM
56
```

```
!0006 ACTR
01
02                                          ; THIS SUBROUTINE PRINTS OUT
03                                          ; AC1 IN OCTAL
04
05
06 00271'054431 CODE:   STA 3,MORE          ; STORE RETURN ADDRESS
07 00272'034431         LDA 3,.5
08 00273'054036-        STA 3,LOAP          ; INIT. LOOP COUNTER
09 00274'125100         MOVL 1,1            ; ROTATE BIT 0 TO CARRY
10 00275'044427         STA 1,TEMP          ; STORE CODE
11 00276'024427         LDA 1,.60           ; ASCII CODE FOR 0
12 00277'125002         MOV 1,1,SZC         ; WAS OLD BIT 0 A ZERO
13 00300'125400         INC 1,1             ;  MAKE AC1 CODE FOR 1
14 00301'004764         JSR OUTPT           ; OUTPUT CODE TO TTO
15 00302'024422         LDA 1,TEMP          ; GET CODE
16 00303'125100         MOVL 1,1            ; INTO PROPER
17 00304'044420         STA 1,TEMP          ; POSITION
18 00305'024417 CODE1:  LDA 1,TEMP          ; RESTORE CODE
19 00306'125100         MOVL 1,1            ; ROTATE AC1
20 00307'125100         MOVL 1,1            ;  3
21 00310'125100         MOVL 1,1            ; LEFT
22 00311'044413         STA 1,TEMP          ; STORE CODE
23 00312'030414         LDA 2,.7
24 00313'147400         AND 2,1             ; MASK OUT TO 3 BITS
25 00314'030411         LDA 2,.60
26 00315'147000         ADD 2,1             ; MAKE CODE ASCII
27 00316'004747         JSR OUTPT           ; OUPUT CODE TO TTO
28 00317'014036-        DSZ LOAP            ; FINISHED YET?
29 00320'000765         JMP CODE1           ; NOT FINISHED
30 00321'002401         JMP @MORE           ; RETURN TO CALL+1
31 00322'000000 MORE:   0
32 00323'000005 .5:     5
33 00324'000000 TEMP:   0
34 00325'000060 .60:    60
35 00326'000007 .7:     7
36
```

```
01
02                                          ; OUTPUT TRAN,RECEPTION ERRORS
03
04 00327'054433 TRANS:  STA 3,LEAVE         ; STORE RETURN ADDRESS
05 00330'014013-        DSZ STOP            ; STOP=1 IS PRINT SUPPRESS
06 00331'000403         JMP .+3             ; NO PRINT SUPPRESS
07 00332'010013-        ISZ STOP            ; YES SUPPRESS OUTPUT
08                                          ; PUT STOP=1 AGAIN
09 00333'002427         JMP @LEAVE          ; RETURN TO MAINLINE
10 00334'063610         SKPDN TTI           ; HAS KEYBOARD BEEN STRUCK
11 00335'000412         JMP OUT             ; NO OUTPUT CODES
12 00336'030425         LDA 2,.177          ; YES
13 00337'064610         DIAC 1,TTI          ; KEYBOARD TO AC1,CLEAR FLAG
14 00340'147400         AND 2,1             ; MASK OUT TO 7 BITS
15 00341'030423         LDA 2,.3
16 00342'146414         SUB# 2,1,SZR        ; IS IT CONTROL P
17 00343'000404         JMP OUT             ; NO, INVALID, CONTINUE OUTPUT
18                                          ; YES CONTROL P
19 00344'126520         SUBZL 1,1           ; TERMINATE PRINTING ERRORS
20 00345'044013-        STA 1,STOP          ; STOP=1
21 00346'002414         JMP @LEAVE          ; RETURN TO MAINLINE
22
23                                          ; NOW STOP HAS BEEN PUT TO -1
24
25 00347'102400 OUT:    SUB 0,0             ; NOW STOP=0 WHICH IS
26 00350'040013-        STA 0,STOP          ; CORRECT FOR NON SUPPRESSION
27 00351'024015-        LDA 1,TEM0          ;
28 00352'004717         JSR CODE            ; OUTPUT TRANSMITTED WORD
29 00353'006000-        JSR @RITER          ; PRINT MESSAGE
30 00354'001135'        PT1
31 00355'024017-        LDA 1,TEM2
32 00356'004713         JSR CODE            ; OUTPUT RECEIVED WORD
33 00357'006000-        JSR @RITER          ; PRINT MESSAGE
34 00360'001150'        PT2
35 00361'002401         JMP @LEAVE          ; RETURN TO MAINLINE
36 00362'000000 LEAVE:  0
37 00363'000177 .177:   177
38 00364'000003 .3:     3
39
```

```
!0008 ACTR
01                                      ; INTERRUPT SERVICE ROUTINE
02
03
04 00365'065477 INT:    DIB 1,CPU      ; GET DEVICE CODE CAUSING INT.
05 00366'030024-         LDA 2,.14      ; 14 = CODE OF RTC
06 00367'146415          SUB# 2,1,SNR
07 00370'000416          JMP SERV1      ; RTC CAUSED INTERRUPT
08 00371'030025-         LDA 2,.40      ; 40 = CODE OF ACTR
09 00372'146415          SUB# 2,1,SNR
10 00373'000434          JMP SERV2      ; ACTR CAUSED INTERRUPT
11 00374'044026-         STA 1,HOLD     ; UNKNOWN INTERRUPT
12 00375'006000-         JSR @RITER     ; WRITE ERROR MESSAGE
13 00376'000666'         MES9
14 00377'024026-         LDA 1,HOLD
15 00400'006023-         JSR @CODER     ; OUTPUT DEVICE CODE
16 00401'063640          SKPDN ACTR     ; IS ACTR DONE=1
17 00402'000403          JMP .+3        ; NO
18 00403'006000-         JSR @RITER     ; WRITE MESSAGE
19 00404'000706'         MES10
20 00405'002001-         JMP @END       ; GO TO ENDUP
21
22
23
24
25                                      ; CLOCK INTERRUPT SERVICE
26                                      ; ROUTINE
27
28 00406'014005-SERV1:   DSZ TAG
29 00407'002032-         JMP @TICK      ; OK CLOCK STABILIZED
30
31 00410'014003-         DSZ TAG2
32 00411'002033-         JMP @MASKD     ; OK INTERRUPT CLEARED
33
34 00412'014004-         DSZ TAG3
35 00413'002031-         JMP @CONTD     ; OK ACTR MASK OPERATIONAL
36
37                                      ; TO REACH HERE, ACTR INTERRUPT
38                                      ; HAS FAILED
39
40 00414'006000-         JSR @RITER     ; PRINT INTERRUPT ERROR
41 00415'000743'         MES5           ; MESSAGE
42 00416'063640          SKPDN ACTR     ; DID TRANSMISSION TERMINATE
43 00417'000403          JMP .+3        ; NO
44 00420'006000-         JSR @RITER     ; NO PRINT MESSAGE
45 00421'000763'         MES6
46 00422'063640          SKPDN ACTR     ; DID RECEPTION OCCUR
47 00423'000403          JMP .+3        ; YES
48 00424'006000-         JSR @RITER     ; NO PRINT MESSAGE
49 00425'001011'         MES20
50 00426'002001-         JMP @END       ; TERMINAL ERROR!!!!!!!!
51
```

```
01
02                                          ; ACTR INTERRUPT SERVICE ROUTINE
03
04 00427'014002-SERV2:  DSZ TAG1
05 00430'002030-        JMP @BACKD          ; OK ACTR INTERRUPT GENERATED
06
07 00431'014005-        DSZ TAG             ; INTERRUPT DURING CLOCK STAB.?
08 00432'000404         JMP .+4             ; NO
09 00433'006000-        JSR @RITER          ; YES WRITE ERROR MESSAGES
10 00434'001160'        MES7
11 00435'002001-        JMP @END            ; TERMINAL ERROR
12
13 00436'014003-        DSZ TAG2            ; INTERRUPT AFTER NIOC?
14 00437'000404         JMP .+4             ; NO
15 00440'006000-        JSR @RITER          ; YES WRITE ERROR MESSAGES
16 00441'000720'        MES13
17 00442'002001-        JMP @END            ; TERMINAL ERROR
18
19                                          ; INTERRUPT HAD TO BE GENERATED
20                                          ; DURING MASK OUT TESTS
21 00443'006000-        JSR @RITER          ; WRITE ERROR MESSAGES
22 00444'001204'        MES8
23 00445'002001-        JMP @END            ; TERMINAL ERROR
24
25
```

```
01
02
03                                          ; TITLES AND ERROR MESSAGES
04 00446'005015 MESSA:  .TXT *<15><12>        ACTR DIAGNOSTICS <15><12>
05 00464'020040         <12>FIRST FLAGS AND INTERRUPTS WILL BE TESTED
06 00515'006411         <15><12><12>*
07 00520'052502 MES1:   .TXT *BUSY FLAG DOES NOT RESPOND TO IORST
08 00542'006411         <15><12>*
09 00544'047504 MES2:   .TXT *DONE FLAG DOES NOT RESPOND TO IORST
10 00566'006411         <15>>12>*
11 00572'041501 MES21:  .TXT *ACTR INTERRUPT GENERATED WHEN DONE=0
12 00614'006411         <15><12>*
13 00616'052502 MES22:  .TXT *BUSY FLAG DOES NOT RESPOND TO NIOC
14 00637'004440         <15><12>*
15 00642'047504 MES23:  .TXT *DONE FLAG DOES NOT RESPOND TO NIOC
16 00663'004440         <15><12>*
17 00666'047125 MES9:   .TXT *UNKNOWN INTERRUPT ,DEV. CODE = *
18 00706'005015 MES10:  .TXT *<15><12>AND ACTR DONE=1<15><12>*
19 00720'041501 MES13:  .TXT *ACTR INTERRUPT NOT CLEARED BY NIOC
20 00741'006411         <15><12>*
21 00743'041501 MES5:   .TXT *ACTR INTERRUPT NOT GENERATED<15><12>*
22 00763'047516 MES6:   .TXT *NON-TERMINATION OF TRANSMISSION (BUSY=1)
23 01007'006411         <15><12>*
24 01011'047510 MES20:  .TXT *HOWEVER TRANSMISSION TERMINATED (DONE=1)
25 01035'006411         <15><12>*
26 01037'020040 MES40:  .TXT *   TRANSFER INFORMATION<15><12>*
27 01054'041440 MES41:  .TXT * CONTROL WORDS TRANSFERRED INCORRECTLY
28 01077'006411         <15><12>*
29 01101'042040 MES42:  .TXT * DATA WORDS TRANSFERRED INCORRECTLY
30 01122'004531         <15><12>*
31 01125'042440 MES43:  .TXT * END OF TEST<15><12>*
32 01135'053440 PT1:    .TXT * WAS TRANSMITTED AND *
33 01150'053440 PT2:    .TXT * WAS RECEIVED<15><12>*
34 01160'041501 MES7:   .TXT *ACTR INTERRUPT NOT CLEARED BY IORST
35 01201'004524         <15><12>*     .
36 01204'041501 MES8:   .TXT *ACTR MASK OUT FAILURE<15><12>*
37 01220'046106 MES31:  .TXT *FLAGS AND INTERRUPT OPERATIONAL
38 01237'020114         <15><12>NOW PROCEEDING WITH TRANSMISSION TESTS
39 01267'020123         <15><12>TO TERMINATE LISTING OF ERROR WORDS
40 01316'006411         <15><12>TYPE CONTROL C
41 01326'020103         <15><12><12>*
42       000000         .NOLOC 0
```

```
!0011 ACTR
01
02                                                              62
03                      .ZREL           ; PAGE 0 CONSTANT STORAGE
04
05 00000-000244'RITER:  WRITE
06 00001-000222'END:    ENDUP
07 00002-000000 TAG1:   0
08 00003-000000 TAG2:   0
09 00004-000000 TAG3:   0
10 00005-000000 TAG:    0
11 00006-000200 .200:   200
12 00007-000000 CT1:    0
13 00010-000000 CT2:    0
14 00011-000377 .377:   377
15 00012-000000 STRPTR: 0
16 00013-000000 STOP:   0
17 00014-000017 .17:    17
18 00015-000000 TEM0:   0
19 00016-000000 TEM1:   0
20 00017-000000 TEM2:   0
21 00020-000000 TEM3:   0
22 00021-000052 STAR:   52
23 00022-000365'INTPTR: INT
24 00023-000271'CODER:  CODE
25 00024-000014 .14:    14
26 00025-000040 .40:    40
27 00026-000000 HOLD:   0
28 00027-077777 NO:     77777
29 00030-000061'BACKD:  BACK
30 00031-000130'CONTD:  CONT
31 00032-000046'TICK: CLOCK
32 00033-000113'MASKD:  MASK
33 00034-000000 TIME:   0
34 00035-000000 SAVE:   0
35 00036-000000 LOAP:   0
36 00037-000004 .4:     4
37
38
39                      .END START

**00000 TOTAL ERRORS, 00000 PASS 1 ERRORS
```

AHEAD	000241'		5/19	5/27					
BACK	000061'		3/12	11/29					
BACKD	000030-		9/05	11/29					
CLOCK	000046'		2/57	11/31					
CODE	000271'		5/03	5/07	6/06	7/28	7/32	11/24	
CODE1	000305'		6/18	6/29					
CODER	000023-		8/15	11/24					
CONT	000130'		4/09	11/30					
CONTD	000031-		8/35	11/30					
CT1	000007-		4/14	4/42	5/06	11/12			
CT2	000010-		4/15	4/50	5/02	11/13			
END	000001-		2/35	2/40	3/16	3/22	3/27	8/20	8/50
			9/11	9/17	9/23	11/06			
ENDUP	000222'		5/10	11/06					
ERR1	000251'		5/39	5/48					
FINIS	000210'		4/21	4/25	4/60				
HOLD	000026-		8/11	8/14	11/27				
INT	000365'		8/04	11/23					
INTPT	000022-		2/22	11/23					
LEAVE	000362'		7/04	7/09	7/21	7/35	7/36		
LOAP	000036-		6/08	6/28	11/35				
LOOP	000140'		4/18	4/48	4/59				
MASK	000113'		3/48	11/32					
MASKD	000033-		8/32	11/32					
MES1	000520'		2/34	10/07					
MES10	000706'		8/19	10/18					
MES13	000720'		9/16	10/19					
MES2	000544'		2/39	10/09					
MES20	001011'		8/49	10/24					
MES21	000572'		3/15	10/11					
MES22	000616'		3/21	10/13					
MES23	000642'		3/26	10/15					
MES31	001220'		4/10	10/37					
MES40	001037'		5/01	10/26					
MES41	001054'		5/05	10/27					
MES42	001101'		5/09	10/29					
MES43	001125'		5/11	10/31					
MES5	000743'		8/41	10/21					
MES6	000763'		8/45	10/22					
MES7	001160'		9/10	10/34					
MES8	001204'		9/22	10/36					
MES9	000666'		8/13	10/17					
MESSA	000446'		2/27	10/04					
MORE	000322'		6/06	6/30	6/31				
NO	000027-		4/11	11/28					
OUT	000347'		7/11	7/17	7/25				
OUTPT	000265'		4/55	5/42	5/46	5/52	6/14	6/27	
PT1	001135'		7/30	10/32					
PT2	001150'		7/34	10/33					
RETUR	000263'		5/34	5/35	5/37	5/41	5/45	5/49	
RITER	000000-		2/26	2/33	2/38	3/14	3/20	3/25	4/09
			4/60	5/04	5/08	5/10	7/29	7/33	8/12
			8/18	8/40	8/44	8/48	9/09	9/15	9/21
			11/05						
SAVE	000035-		2/21	5/14	11/34				
SERV1	000406'		8/07	8/28					
SERV2	000427'		8/10	9/04					
STAR	000021-		4/54	11/22					
START	000000'	EN	2/04	2/14	11/39				

STOP	000013-		4/16	4/51	4/56	7/05	7/07	7/20	7/26
			11/16						
STRPT	000012-		4/12	4/18	4/19	4/22	4/23	11/15	
TAG	000005-		2/42	2/60	3/31	3/54	8/28	9/07	11/10
TAG1	000002-		2/44	2/58	3/32	3/53	9/04	11/07	
TAG2	000003-		2/45	3/01	3/29	3/55	8/31	9/13	11/08
TAG3	000004-		2/46	3/02	3/33	3/51	8/34	11/09	
TASK	000401	NC	2/06						
TEM0	000015-		4/35	4/44	7/27	11/18			
TEM1	000016-		4/36	4/43	11/19				
TEM2	000017-		4/37	4/46	7/31	11/20			
TEM3	000020-		4/38	4/45	11/21				
TEMP	000324'		6/10	6/15	6/17	6/18	6/22	6/33	
TICK	000032-		8/29	11/31					
TIME	000034-		2/18	5/17	5/21	11/33			
TRANS	000327'		4/41	4/58	7/04				
WRITE	000244'		5/34	11/05					
.14	000024-		8/05	11/25					
.17	000014-		4/26	11/17					
.177	000363'		7/12	7/37					
.200	000006-		3/48	11/11					
.3	000364'		7//15	7/38					
.377	000011-		5/38	11/14					
.4	000037-		5/20	11/36					
.40	000025-		8/08	11/26					
.5	000323'		6/07	6/32					
.60	000325'		6/11	6/25	6/34				
.7	000326'		6/23	6/35					
.PT	000264'		5/36	5/39	5/47	5/50			

R

BIBLIOGRAPHY

ECLIPSE- Line Real Time Operating System User's Manual Rev. ØØ. Southboro Mass.: Data General Corporation, 1975.

ECLIPSE- Line Real Time Disc Operating System User's Manual Rev.Ø1. Southboro Mass.: Data General Corporation, 1975.

How to Use Nova Computers Rev. Ø9. Southboro Mass.: Data General Corporation, 1974.

Introduction to ECLIPSE- Line Real Time Operating System Rev.ØØ. Southboro Mass.: Data General Corporation, 1975.

Introduction to ECLIPSE- Line Real Time Disc Operating System Rev. ØØ. Southboro Mass.: Data General Corporation, 1975.

User's Manual ECLIPSE Macro Assembler Rev. ØØ. Southboro Mass.: Data General Corporation, 1975.

Programmer's Reference Manual, ECLIPSE Line Computers Rev. Ø4. Southboro Mass.: Data General Corporation, 1975.